



universität
wien

Dissertation / Doctoral Thesis

Titel der Dissertation / Title of the Doctoral Thesis

Efficient Machine Learning Methods for Time Series with Applications to Trip Reconstruction in Mobility Research

verfasst von / submitted by
Maximilian Leodolter

angestrebter akademischer Grad / in partial fulfillment of the requirements for the degree of
Doktor der Technischen Wissenschaften (Dr. techn.)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt /
degree programme code as it appears on
the student record sheet:

UA 786 880

Studienrichtung lt. Studienblatt /
degree programme as it appears on
the student record sheet:

Informatik

Betreut von / Supervisor:

Univ.-Prof. Dr. Claudia Plant

Betreut von / Supervisor:

DI Dr. Norbert Brändle

Abstract

This thesis proposes novel algorithms for time series mining specifically designed for trip reconstruction, which is a subdomain of mobility research. The proposed methods advance established methods from machine learning and time series mining to efficiently retrieve and discover patterns in sensor data, especially accelerometer and GPS data. These patterns facilitate a higher level understanding of the dynamics and processes generating the collected mobility data, for example the chosen mode of transportation. This understanding is crucial to evaluate policy initiatives, make informed decisions to adjust traffic-regulation measures, and ultimately achieve higher goals such as reducing traffic-induced air pollution.

Although the proposed methods are designed for analysing mobility data, the algorithms are also beneficial for time series data from different domains, as the accompanying research papers, that present the developed methods, also demonstrate.

Time series data are ubiquitous across many fields, not only mobility research, but also finance, medicine and speech recognition. The overwhelming amount of data opens up many opportunities, but also poses the challenge of developing algorithms that can process the amount of data, often even in real time. The methods proposed in this thesis can help gaining high level knowledge from data to improve business processes, make informed management decisions or recognise critical situations early.

Zusammenfassung

Diese Doktorarbeit präsentiert neuartige Algorithmen für das Analysieren von Zeitreihen, die speziell für die Rekonstruktion von Transportwegen entwickelt wurden, einem Teilbereich der Mobilitätsforschung. Die vorgeschlagenen Methoden bauen auf etablierten Verfahren aus den Bereichen Machine Learning, Data Mining und Zeitreihenanalyse auf, um Muster in Sensordaten, insbesondere Accelerometer- und GPS-Daten, effizient zu erkennen. Diese Muster ermöglichen ein besseres Verständnis der Prozesse, die den gesammelten Mobilitätsdaten zu Grunde liegen, beispielsweise das Transportmittel. Dieses Verständnis kann das Evaluieren politischer Initiativen unterstützen, sowie informierte Entscheidungsfindungen zur Anpassung von Verkehrsregelungen und letztendlich das Erreichen höherer Ziele wie die Reduzierung von verkehrsbedingter Luftverschmutzung.

Obwohl die vorgeschlagenen Methoden für die Analyse von Mobilitätsdaten konzipiert sind, sind die Algorithmen auch für Zeitreihendaten aus anderen Domänen vorteilhaft, wie die wissenschaftlichen Publikationen auf welchen diese Doktorarbeit aufbaut und dieser angehängt sind, zeigen.

Zeitreihendaten sind in vielen Bereichen allgegenwärtig, nicht nur in der Mobilitätsforschung, sondern auch in der Finanzwelt, der Medizin und der Spracherkennung. Die überwältigende Datenmenge eröffnet viele Möglichkeiten, birgt jedoch auch die Herausforderung, Algorithmen zu entwickeln, die die Datenmenge verarbeiten können, oft sogar in Echtzeit. Die in dieser Doktorarbeit vorgeschlagenen Methoden können dazu beitragen, Erkenntnisse aus Daten zu gewinnen, um Geschäftsprozesse zu verbessern, informierte Managemententscheidungen zu treffen oder kritische Situationen frühzeitig zu erkennen.

Acknowledgements

First of all I would like to thank my supervisor Claudia, who guided me through this journey and also gave me the chance to work together with her team. Her critical and result-oriented thinking shaped my way of working.

I want to express my special gratitude to the reviewers of this thesis, Prof. Leisch from the University of Natural Resources and Life Sciences, Vienna, and Ao. Prof. Rauber from the Technical University of Vienna.

Without the support from the Austrian Institute of Technology this work wouldn't have been possible. Further I want to thank all the colleagues from AIT, especially Peter, Christian, Anita, Hannes, Melitta, Florian and Markus. First and foremost, I would like to thank Norbert, whose support, countless fruitful conversations and valuable feedback have been so important to me over these years. I would also like to thank the colleagues of the data mining team at the University of Vienna, especially Ben, Martin and Lukas, for all the hours in which we exchanged ideas or shared the ups and downs of a PhD student's life.

My family supported and encouraged me throughout my academic career. I am deeply grateful.

My very special thanks go to Mira, who was always there for me, especially during the frustrating times, and motivated me to finish this project.

This dissertation was a lifelong project, I want to thank everyone who contributed knowingly or unknowingly.

This journey wasn't always easy. I wouldn't want to miss it.

Publications

Most of the results of this thesis were already published in conference proceedings and journal articles. Therefore, this thesis is based on the following publications. My contributions of the following papers comprise the core work of development, implementation and mathematical derivations and drafting/writing of the paper. The co-authors contributed to publications via guidance, supervision and editing of the paper text.

- **Paper A:** Maximilian Leodolter, Peter Widhalm, Claudia Plant, Norbert Brändle, "Semi-supervised Segmentation of Accelerometer Time Series for Transport Mode Classification", 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS). IEEE, 2017.
- **Paper B:** Maximilian Leodolter, Norbert Brändle, Claudia Plant, "Automatic Detection of Warped Patterns in Time Series: The Caterpillar Algorithm", 2018 IEEE International Conference on Big Knowledge (ICBK). IEEE, 2018.
- **Paper C:** Maximilian Leodolter, Claudia Plant, Norbert Brändle, "IncDTW: An R package for incremental calculation of dynamic time warping", *Journal of Statistical Software* 99 (2021): 1-23.
- **Paper D:** Maximilian Leodolter, Claudia Plant, Norbert Brändle, "Rotation Invariant GPS Trajectory Mining", *GeoInformatica* (2023): 1-27.

Papers written during the time of my PhD which are beyond the scope of this dissertation:

- **Paper E:** Widhalm, Peter, Maximilian Leodolter, and Norbert Brändle. "Top in the lab, flop in the field? Evaluation of a sensor-based travel activity classifier with the SHL dataset." *Proceedings of the 2018 ACM international joint conference and 2018 international symposium on pervasive and ubiquitous computing and wearable computers*. 2018.
- **Paper F:** Feldbauer, Roman, Maximilian Leodolter, Claudia Plant, Arthur Flexer, "Fast approximate hubness reduction for large high-dimensional data." 2018 IEEE International Conference on Big Knowledge (ICBK). IEEE, 2018.
- **Paper G:** Widhalm, Peter, Maximilian Leodolter, and Norbert Brändle. "Ensemble-based domain adaptation for transport mode recognition with mobile sensors." *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and*

Proceedings of the 2019 ACM International Symposium on Wearable Computers. 2019.

- **Paper H:** Widhalm, Peter, Maximilian Leodolter, and Norbert Brändle. "Into the wild—avoiding pitfalls in the evaluation of travel activity classifiers." *Human activity sensing: corpus and applications (2019)*: 197-211.

During my PhD I also developed and published the open sources software package IncDTW, which is listed on CRAN. Paper J is an extended version of Paper C, and is published without peer review as software package documentation for the software package.

- **Paper I:** Maximilian Leodolter, Norbert Brändle, Claudia Plant, "IncDTW: an R package for Incremental Dynamic Time Warping", 2018, eRum 2018, Poster
- **Paper J:** Maximilian Leodolter, Claudia Plant, Norbert Brändle, "Theory and Applications for the R-Package IncDTW", 2017, <https://CRAN.R-project.org/package=IncDTW>,
- **Software:** Maximilian Leodolter "IncDTW: Incremental Calculation of Dynamic Time Warping", R package version 1.1.4, 2017, <https://CRAN.R-project.org/package=IncDTW>

Contents

Abstract	ii
Zusammenfassung	iv
Acknowledgements	vi
Publications	viii
Contents	x
1 Introduction	1
1.1 Research Questions	4
2 Data Mining and Machine Learning	6
2.1 Supervised Learning	6
2.2 Unsupervised Learning	7
2.3 Semi-supervised Learning	9
3 Machine Learning Methods for Time Series Data	11
3.1 Distance Measures	13
3.2 Time Series Clustering	15
3.3 Pattern Recognition and Sliding Algorithms	16
4 Challenges and Approaches from Mobility Research	19
4.1 Trip Reconstruction	19
4.2 Recognition of Mobility Patterns	23
4.3 Runtime Efficient Algorithms	24
5 Contributions, Conclusion and Perspectives	26
5.1 Contributions	26
5.2 Conclusion and Discussion	27
5.3 Outlook	29
Bibliography	30

Publications	36
A Semi-supervised Segmentation of Accelerometer Time Series for Transport Mode Classification	37
B Automatic Detection of Warped Patterns in Time Series: The Caterpillar Algorithm	44
C IncDTW: An R Package for Incremental Calculation of Dynamic Time Warping	54
D Rotation Invariant GPS Trajectory Mining	78

Chapter 1

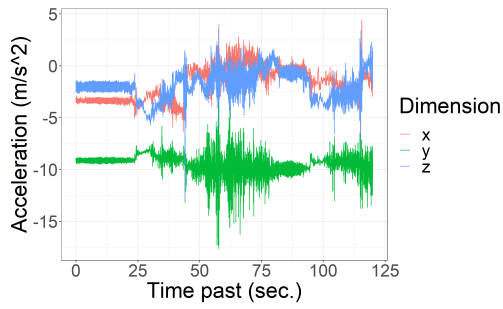
Introduction

Innovations and new technologies in recent decades have enabled the modern world to be shaped by measuring, monitoring and optimizing processes and decisions across many fields of business and research. Massive amounts of data is both a driving force but also a result of this evolution. One area where these developments have had a profound impact is in analysing time series, which is the collection of sequential observations. Machine learning methods to analyse time series and gain knowledge out of such analyses, find application in various domains, including finance, medicine, weather forecasting, and mobility research.

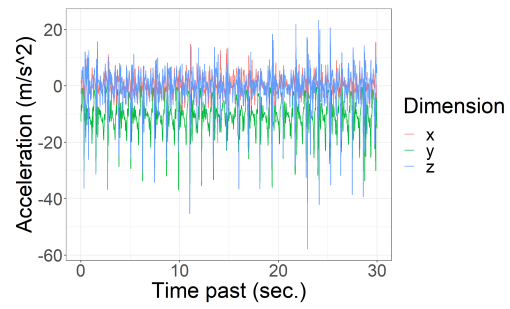
The field of mobility research investigates the dynamics, motivation, structure, and outcomes associated with the movement of people and vehicles or other objects. In the realm of mobility research, the collection and analysis of time series data are also important. Time series data in mobility research can, for example, contain the sequentially observed information about the status, position or speed of a moving object. Figure 1.1 gives an example of such a sequence, by visualizing the GPS and accelerometer signals recorded by a person riding the bus or walking. The analysis and derivation of high level information of such data is part of the research field of trip reconstruction. Moreover, trip reconstruction [8] is a subdomain of mobility research and deals with the identification of the start, end, segmentation, intermediate states and transport modes of a trip by the help of data analysis.

This thesis is a presentation of machine learning methods for time series, that help to better understand the data generating process behind the time series, with special focus on data for trip reconstruction. Specific actions or movements along a transportation trip may cause specific patterns in sensor recordings. Recognising these patterns and assigning them to the correct cause is crucial for a comprehensive understanding of the chain of events taking place along a transportation trip.

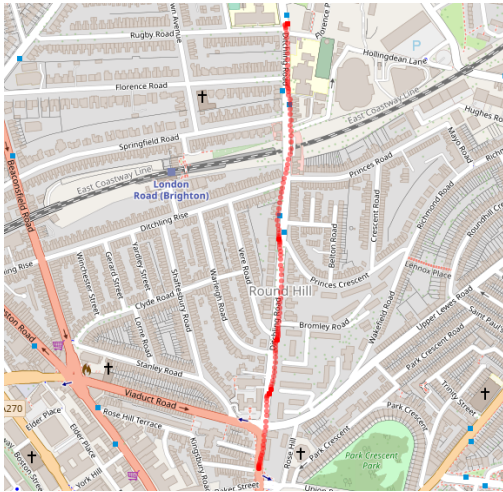
One of the biggest challenges in trip reconstruction is the manifoldness of patterns that could describe the same root cause. This is due to the fact that the movement causing the pattern can vary along multiple dimensions, e.g. the person or vehicle that executes the movement, speed and duration of the movement, position or orientation of the recording device, and external factors such as traffic situation. For this reason,



(a) Acceleration signal of a bus trip segment



(b) Acceleration signal of a walk trip segment



(c) GPS signal of a bus trip segment



(d) GPS signal of a walk trip segment

Figure 1.1: Sensor recordings of a smartphone attached to a person's hip while walking or riding the bus.

we developed algorithms for pattern recognition while respecting these variations. The algorithms Seg-IP and Segment-123 (Paper A) help to adjust erroneous user-provided input and build a semi-supervised transport mode classifier model. The Caterpillar algorithm (Paper B) is a combination of the Minimum Description Length (MDL) principle and the distance measure Dynamic Time Warping (DTW), and is capable of detecting time series patterns that vary in speed. In addition, the S3KR algorithm (Paper D) recognizes patterns in time series independent of any rotation, scale and position. Hence the algorithm S3KR detects similarities in GPS trajectories regardless of the orientation and/or speed and/or position in which a person or vehicle is moving.

Apart from the complexity of detecting patterns accurately, algorithms dealing with this type of time series data also need to work efficiently. Runtime efficiency is necessary, on the one hand because applications based on data analysis algorithms often run live and the data must therefore be processed in real time. On the other hand because efficient algorithms allow more processing in the same amount of time and can therefore produce more accurate results. Furthermore, efficient algorithms require less computation power,

and so less energy consumption, and consequently produce less greenhouse gas emissions.

To meet this challenge, the proposed methods presented in this work are designed to work most efficiently. In addition, we have developed the software package IncDTW that implements the incremental computation of DTW – applied in the Caterpillar algorithm – to update computations for newly arriving observations as efficiently as possible. Paper C elaborates the theoretical foundation for IncDTW. Further, IncDTW serves as toolkit for other researchers to build time series analysis algorithms based on the incremental calculation of DTW.

The remainder of this chapter gives a more detailed introduction in the motivation of mobility research, before this chapter concludes with the research questions this thesis deals with, and an overview of the remainder of this thesis.

Mobility Research

One of the major challenges in mobility research is to better understand mobility behaviour. Based on a better understanding, one can learn how to nudge or change mobility behaviour towards reaching higher goals. These goals are for example the reduction of pollution of green house gas emissions (GHG) and noise emissions in urban areas, or a higher level of efficiency in freight transport.

The motivation for these goals are well known: The European Green Deal [17] says that "In 2018, the transport sector, including aviation, was responsible for one quarter of emissions in the EU27. To achieve the objective of net carbon neutrality by 2050, emissions in the transport sector need to be reduced by 90%. In December 2020, the Commission proposed a Sustainable and Smart Mobility Strategy. The strategy is based on three pillars: 1) Make all modes of transport more sustainable, 2) Make sustainable alternatives widely available in a multi-modal transport system, and 3) Put in place the right incentives to drive the transition"

And further the World Energy Outlook states [27] that: "Emissions trends in the transport sector are determined by how quickly oil can be displaced; at present it accounts for 90% of energy use in transport. Both passenger and freight activity are set to more than double by 2050, underpinned by higher mobility demand needs in the developing world as economies and populations expand and living standards rise. Reductions in emissions of around one-quarter to 2030 in the NZE Scenario are driven by increased electrification, efficiency improvements and behaviour change."

According to both reports one key strategy is to make alternative low-emission transport means more attractive and to target a behaviour change of passengers towards a multi-modal mobility behaviour based mainly on sustainable transport means. Achieving a behaviour change also requires the capability to measure the mobility behaviour. This is already possible for many areas in the transportation sector, however to measure the modal split of passengers remains a burdensome challenge. Until recently the state of the art method has relied on pen and paper interviews which used to be cost intensive and resulted in coarse data [42]. During the last few years these surveys have been partly replaced by smartphone-based systems that recreate multimodal trips of individ-

uals. This new approach has the potential to gather more fine-grained and longitudinal mobility data with significantly less effort required from the participants [42], [15]. This PhD thesis puts a special emphasis on fundamental research and the development of new pattern matching algorithms, applicable for time series data of plenty of different domains. Besides the purpose of contributing to the research field of time series data mining in general, this work was also motivated to improve an already existing system – for gathering smartphone supported travel-diary data – at the AIT (Austrian Institute of Technology).

1.1 Research Questions

This section states the two main research questions (RQ) and sub-questions addressed in this thesis. The following enumeration gives a guidance which section of this thesis introduces the concepts and research fields behind the research questions, and which of the published research papers of this thesis proposes solutions to the respective research questions.

Research Topic 1: How can machine learning methods help to retrieve and detect prototypical patterns in time series data, especially for the use case of trip reconstruction?

- **RQ 1.a:** How can machine learning models incorporate erroneous user-feedback in a semi-supervised fashion to improve the overall quality of a transport mode detection system?
Section 4 gives more details about mobility research and Paper A addresses this question by proposing the algorithms Seg-IP and Segment-123.
- **RQ 1.b:** How to detect transport mode specific patterns of varying lengths in accelerometer time series?
Section 3 gives more details about machine learning methods for time series and Paper B addresses this question by proposing the Caterpillar algorithm.
- **RQ 1.c:** How to detect patterns in GPS trajectories independently of any rotation?
Section 3 gives more details about machine learning methods for time series and Paper D addresses this question by proposing the algorithm S3KR.

Research Topic 2: How can machine learning methods operate most efficiently to enable the analysis of time series data in real-time, or to enable the detailed analysis of big databases of time series in manageable time?

- **RQ 2.a:** How to efficiently retrieve and detect prototypical patterns in a set of time series of different lengths?

Section 3.3 elaborates sliding algorithms for time series and Paper B and D address this question by presenting algorithms that analyse time series in a sliding fashion.

- **RQ 2.b:** How to most efficiently apply DTW in a sliding algorithm to detect patterns of varying lengths?

Section 3.1 introduces the distance measure DTW. Paper B addresses this question, while Paper C presents the fundamental work for achieving efficiency.

The remainder of this thesis is structured as follows: Chapter 2 gives a basic introduction to the areas of data mining and machine learning covered in this work. Chapter 3 discusses some peculiarities of data mining methods for time series data that are relevant for this thesis. Chapter 4 introduces briefly the field of trip reconstruction in mobility research. Chapter 5 deals with the conclusion, discussion and prospects for future work that could be based on this thesis. The published research papers on which this work is based are included in the appendix.

Chapter 2

Data Mining and Machine Learning

Data mining and machine learning are both fields within the broader realm of computer science that deal with analyzing and extracting insights from data. Both data mining and machine learning rely heavily on statistical analysis and algorithms, and they share many techniques and methods. While data mining is often focused on the discovery of patterns in historical data, machine learning is more concerned with using that data to make predictions and automated decisions in real-time.

The following sections only give a brief introduction to the most important concepts of data mining and machine learning required to understand all concepts of this thesis. A more detailed and thorough introduction in data mining and machine learning methods can be found in [24].

2.1 Supervised Learning

Supervised Learning is the exercise to learn a relation between a target variable Y (in literature also often called dependent variable or response) and the feature variables X (also called independent variables or inputs). If Y is a categorical variable that can only take values of a limited set, for example ('car', 'bicycle', 'train') then we speak of a *classification* problem. On the other hand, if Y is a continuous numerical variable, as for example stock prices or the estimated driving time for a trip, then we speak of a *regression* problem. The inputs X represent observations and can also take any form, either numerical, or categorical. An input could be the number of cylinders or the manufacturer of a vehicle.

Having a model $f(\cdot)$ that describes Y dependent on X plus an error term ϵ

$$Y = f(X) + \epsilon \tag{2.1}$$

one can use the model to derive estimates for Y given X : $\hat{Y} = f(X)$. Finally we can evaluate the model by measuring the error term $(Y - \hat{Y})$. Minimizing the error term

helps the model to predict the target variable more accurately. For this reason this problem category is called supervised, since the observed target variable Y serves as a form of teacher that helps to develop and improve the model. The data set consisting of historical observations used to learn the relation $f(\cdot)$ is called the training data.

The literature [24] gives more details about different supervised learning models. The following two models described briefly give an excerpt of the most elementary but also very popular models. These two models differ fundamentally in the way they model the function $f(\cdot)$ from (2.1), but both build the foundation for state of the art supervised learning models, and serve here as examples how supervised models work.

- **Linear** models make strong assumptions on the data, but deliver consistent estimations and are capable of extrapolation outside of the so-far observed space. Linear models are typically trained by minimizing the estimation error. The most generic linear model is of the form $Y = \beta_0 + X\beta_1 + \epsilon$, so the relation between the target variable Y and the independent variable X is modelled via the linear combination with the parameters β .
- **k-Nearest Neighbor** models (kNN) are non-parametric, in the sense that they make no assumptions on the data. A kNN classifier works by comparing the distance of a new observation to all observation of a training set of historical observations, and forming the estimation via majority vote out of the labels of the k closest neighbors. This aggregation step can be unweighted, or weighted dependent on the distances. A kNN regression model works similarly, but in the aggregation step the target values are aggregated instead of the majority vote of the labels. Section 3.1 details the concept of distances for machine learning and data mining methods. In the literature [13] you can also find variations of the kNN, e.g. Parzen Windows where predictions are based on all neighbors in a neighborhood of predefined size (rather than number of neighbors), or the Convex Containment Method. For all of these nearest neighbor models the choice of the distance measure is key.

Both model categories share the benefit of being easy to interpret and explainable. Also, both models find application for regression as well as classification problems.

2.2 Unsupervised Learning

Unsupervised Learning deals with the problem to find structure in data where no structure is obvious. It allows for the discovery of patterns and relationships in data without the need for explicit labels or guidance. This is particularly useful in situations where the data is complex or high-dimensional, and it may be difficult or impossible for a human expert to manually label every example in the dataset. Unlike to supervised learning, there is no 'teacher' in an unsupervised learning problem. Hence there is no target variable, which a model should describe best by observed features, but rather the

model is designed to find structure in the features directly. Discovering such structure can help users to better understand the data and support decision making.

Apart from the capability of unsupervised learning algorithms in an isolated setting, they are also often applied to preprocess the data before a supervised learning model is applied.

The following sections focus on two unsupervised learning methods that are most important for a full understanding of this thesis: clustering and the Minimum Description Length principle. A more detailed discussion of unsupervised learning can be found in [12].

Clustering

Clustering a set of data means to discover inherent groups of data points, such that the data points inside of one group are most similar and that the data points of any two different groups are most dissimilar. The definition of the similarity measure, or analogously the definition of the distance measure, is key. Section 3.1 elaborates distance measures generally, and specifically for time series. Apart from the distance measure between two data points, next important is how the similarity between a two groups of data points is understood. Hence there is no single truth for this question, and different use cases require different definitions of similarity. There is a wealth of clustering algorithms in the literature. The most important ones to mention here are:

- Centroid-based clustering: The distance of a data point and a group of data points is defined by the distance of that point to the barycenter of the group. This way the data points are assigned to the closest of the groups. The most popular algorithms of this class are k-Means and PAM (Partitioning Around Medoids) [23] and [1].
- Density based clustering: The main idea is that clusters are areas of density, and that clusters are separated by sparsity. This means also, that the assignment of a data point to a cluster relies on the direct neighborhood of that data point, and whether in that neighborhood is enough density of a cluster. The assignment step is independent of the barycenter of the group, and consequently non-spherical clusters can be discovered. Also, contrary to the centroid based clustering, not all data points must be assigned to clusters. If the direct neighborhood of a data point is too sparse, then this point is not assigned to any cluster. The most popular density based clustering algorithm is DBSCAN [16].
- Hierarchical clustering: The algorithm is based on the pairwise distances between all pairs of data points. Starting with all observations interpreted as standalone clusters, a hierarchy of clusters is build by merging iteratively in a greedy fashion the closest clusters together to form a new cluster. The other way round is also possible, to start with all observations in a single cluster, and split those most dissimilar. The choice of the distance for single observations plays a major role, and also how to define the distance between two clusters. [29]

- Spectral clustering is, as hierarchical clustering, also based on the similarity matrix of all observations. The algorithm first reduces the dimensionality via deriving the most relevant dimensions from the spectrum (the eigenvalues) of the similarity matrix. And second, the reduced problem is clustered, for example with k-Means [10].

Most of these clustering methods found application in the research work of this thesis: the algorithm TSS (presented in Paper D) is based on spectral clustering, Segment-123 (Paper A) incorporates PAM, we also applied PAM for the evaluation of the novel distance metric (Paper D), and we applied hierarchical clustering to demonstrate some functionalities of the IncDTW package in the extension paper [36] of Paper C.

The Minimum Description Length (MDL) Principle

The MDL principle can be considered as a formalization of Occam’s Razor, which says if you have two explanations for one and the same problem, that are equally well in describing the problem, then the simpler model – the one that requires fewer assumptions – is the one to select. [22] says: ”The minimum description length (MDL) principle is a powerful method of inductive inference, the basis of statistical modeling, pattern recognition, and machine learning. It holds that the best explanation, given a limited set of observed data, is the one that permits the greatest compression of the data. MDL methods are particularly well-suited for dealing with model selection, prediction, and estimation problems in situations where the models under consideration can be arbitrarily complex, and overfitting the data is a serious concern.” We made use of this principle in Paper B to design an algorithm which decides whether one observation can be understood as a similar representation of another observation, if the compression rate of the differences are smaller than representing these two observations independently.

2.3 Semi-supervised Learning

Semi-supervised Learning [62] can be understood as the research field where supervised and unsupervised learning overlap. Supervised learning relies on historical observations of the target variable Y . There are situations where Y is not always observed for each observation of X . This can be due to the lack of resources (labelling data by human supervision can be time and cost intensive), or because of evolving data gathering methods (in past surveys Y was maybe not part of the survey). In this sense there are similarities between semi-supervised learning and the imputation of missing data.

Say a dataset Ω consists of two disjunct sets where the target variable Y was observed in one of the two sets, and unobserved in the other: $\Omega = \left((X_o, Y_o), (X_u, Y_u) \right)$ and $Y_u =$ missing. The simple approach would be to train the supervised learning models limited to data where Y was observed, so X_o and Y_o . However, semi-supervised learning methods make it possible to gain knowledge out of X_u as well. The main idea is to improve the overall performance of a model by iteratively training on (X_o, Y_o) , predicting the target

variable Y_u for selected observations of X_u , and feeding these enriched observations back to the training data for the next iteration. We make use of this principle in Paper A. More details about semi-supervised learning can be found in [63].

Chapter 3

Machine Learning Methods for Time Series Data

Machine learning methods applied to time series data need to factor in the special characteristics of data structured as time series. This chapter introduces some of these specialties that are necessary to comprehend the Papers A to D and the contributions of this thesis.

A time series x is a sequence of data points x_t for the time $t \in T$. An observation x_t can be univariate or multivariate. Say $x \in \mathbb{R}^{T \times M}$:

$$x = \begin{pmatrix} x_{t,1} & x_{t,2} & \dots & x_{t,M} \\ x_{t+1,1} & x_{t+1,2} & \dots & x_{t+1,M} \\ \dots & \dots & \dots & \dots \\ x_{T,1} & x_{T,2} & \dots & x_{T,M} \end{pmatrix} \quad (3.1)$$

An example for a univariate time series is the ECG (heart rate) of a patient, sampled multiple times per second, and an example for a multivariate time series could be the unemployment rates of all the European countries for the last twenty years, sampled quarterly. Another example would be the bid prices of all the stocks listed in the Dow Jones index for a the last 7 days, sampled every hour. Typically time series are collected in regular intervals. If not, a common practice is to interpolate the observations such that the observations are re-sampled equidistantly. Time series data can origin from many different domains, such as finance, economy, meteorology, health sector, manufacturing, and many more. This thesis deals especially with two types of time series, accelerometer data, and GPS trajectory. Accelerometer time series are typically collected by 3-dimensional sensors, where each data point gives the acceleration at time t of the sensor in each of the three directions in the 3-dimensional space (forward-backward, left-right, up-down). GPS trajectories are sequences of GPS (Global Positioning System) coordinates that are recorded over time by a GPS device or application. GPS records consist of latitude, longitude, and sometimes altitude of a user's location at regular intervals, creating a series of points that represent the user's movement over time – also

called GPS trajectory. Figure 1.1 shows sensor readings (recorded at a frequency of $\tilde{100}$ Hz) of a smartphone attached to a person’s hip, while the person is walking or riding the bus¹. The Figures 1.1c-1.1d show GPS trajectories segments lasting a couple of minutes, extracted from a multiple-hour trip record. The Figures 1.1a-1.1b show the accelerometer readings of some seconds extracted from the GPS segments. The usual presentation of GPS trajectories is embedded in the two dimensional space with the map context, whereas all of the three dimensions of the accelerometer signal are plotted in parallel against the time. The figures show that the characteristics of these signals vary. While the accelerometer signal of a walk has higher variation, it seems that the signal from the bus ride is separated in segments with higher and lower variation. Also the GPS signals shows differences. The bus ride follows a clear pattern, while it seems that the GPS positions of the walk jump more often to the sides of the actual (most likely) walking path. This small example already reveals some of the challenges related to these types of data.

As described in [47], methods for storing and processing times series data has become more and more relevant in recent years. Further [47] list the following popular use cases for processing time series data:

- Indexing: Having a set of time series X , and another time series x , find the time series in X that is closest to x . The task is similar to a k-Nearest Neighbor search, where k is 1.
- Clustering: Find groups of similar time series in a set of time series. See also Section 2.2 and 3.2
- Classification: Given an unlabeled time series x , and a set of available labels, find the label that is most suitable for x . See also Section 2.1.
- Forecast: Given the observations of a time series x until time t , predict the value of x_{t+k} for any time step $k > 0$. This falls in the domain of classical time series analysis with the well known ARIMA, SARIMA, and SARIMAX models [6].
- Summarization: This field deals with the challenge of finding a representation of a very long time series x to better understand x and visually represent it more compact.
- Anomaly Detection: Detecting sections of a time series x that are different to the remaining part of x , or in some other interpretation of special interest.
- Segmentation: Given a time series x with observations ranging from x_1 to x_T , then the challenge is to find single observations x_t , where to split x in homogeneous segments.

¹The data is a sample out of the University of Sussex-Huawei Locomotion and Transportation Dataset [21].

- Representation: Similar to segmentation, the task is to find a representation of x via segmentation and aggregation that is somehow different than x (for example smaller in bit size for storage), but also represent x as close as possible.

Most of these challenges require algorithms, data mining methods or machine learning methods that directly or implicitly apply distance /similarity measures. Two distance measures that are popular in the literature [41, 2, 60] for times series mining in general, but also for the application on accelerometer data and GPS data are the Euclidean distance metric (d_E) and the Dynamic Time Warping (DTW or d_{DTW}) distance measure. For this reason this thesis puts a special focus on these two distance measures and developed algorithms that help to efficiently calculate the distance measures, and/or modify the distance measure for special problems. Hence, the following sections will give a brief introduction and discuss shortly the advantages and disadvantages of d_E and DTW. More details about further time series distances can be found in [37] and [54].

3.1 Distance Measures

Euclidean Distance Metric

For two time series x and y of the same length, the Euclidean distance metric is defined as:

$$d_E(x, y) = \sqrt{\sum_{t=1}^T (x_t - y_t)^2} \quad (3.2)$$

d_E is a metric, so for any time series x , y and z of identical dimensions it fulfills the following conditions:

$$\begin{aligned} d_E(x, y) &= d_E(y, x) \\ d_E(x, y) &\geq 0 \quad \text{and} \quad d_E(x, y) = 0 \Leftrightarrow x = y \\ d_E(x, y) &\leq d_E(x, z) + d_E(z, y) \end{aligned} \quad (3.3)$$

The third condition is the triangle inequality condition, which is used in the literature to speed up algorithms that require multiple distance calculations in a metric space, as the k-nearest neighbor search [55, 9, 44] or clustering [14, 34].

Dynamic Time Warping

The Dynamic Time Warping algorithm – originally introduced by [49] – calculates the distance between two time series x and y by finding the best non-linear alignment of the single observations of x and y . This non-linear alignment is also the main advantage of DTW compared to the Euclidean distance which forces a 1-1 alignment, hence one

observation of x is aligned to exactly one observation of y . The 1-many alignment of DTW allows to detect similar patterns in two time series, that were collected at shifted time periods, and can also have differing extensions in time. Figure 3.1 illustrates this principle. The dashed lines connect the observations of x and y according to the enforced alignment by d_E (a), and the alignment calculated by the DTW algorithm (b).

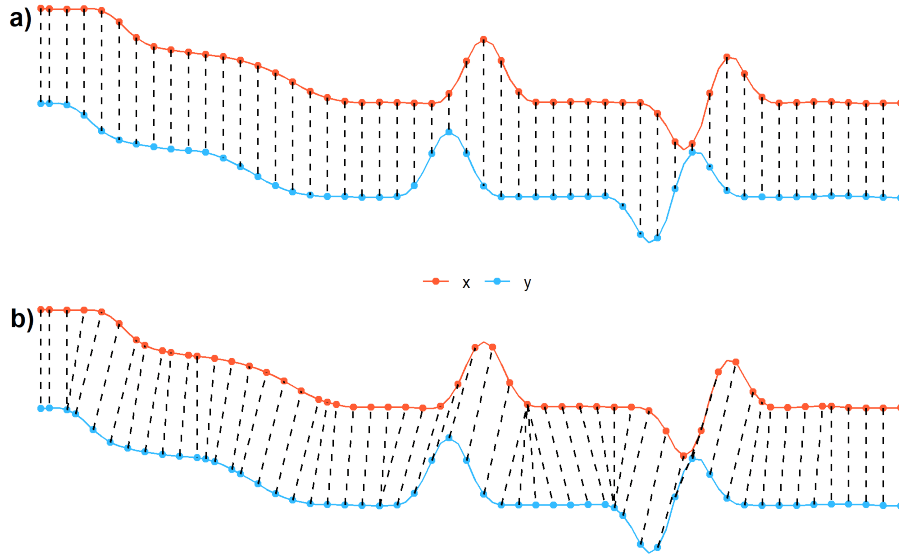


Figure 3.1: Alignments of the Euclidean Distance (a) and the Dynamic Time Warping distance measure (b).

DTW computes the distance of two time series as the accumulated sum of distances of pairs of observations. To find the sequence of pairs that result in the smallest overall distance, the DTW algorithm proceeds as follows. A cost matrix of local distances of all pairwise combinations of single observations is computed. Then these local costs are aggregated by following defined rules to a global cost matrix. And finally the algorithm steps through the global cost matrix in reverse order to find the alignment path (which defines the pairs of observations in Figure 3.1b) with the cheapest overall costs.

The possibility of this procedure to detect similarities of time-shifted time series comes at a certain cost, an increased runtime complexity. This is a disadvantage of the DTW distance measure compared to d_E . While d_E has linear complexity in the number of observations $O(T)$, the complexity of computing d_{DTW} is $O(T^2)$. There are methods to restrict the possible alignment paths for the calculation of d_{DTW} , as for example the Sakoe-Chiba warping window ω [49], which restricts the algorithm to align observations x_t and y_s if $|t-s| > \omega$. This alignment restriction reduces the complexity to $O(T \times \omega)$. Nevertheless, increasing the runtime efficiency of algorithms applying DTW is a popular topic in the literature [50, 3, 32]. This thesis also deals with the research question how to accelerate the computation of DTW, and proposed the implementation of the incremental calculation of DTW, and the vector-based implementation of the cost matrices, see Paper B and C.

A second disadvantage is that DTW does not fulfill the triangle inequality. Again, this increases the required runtime for some data mining tasks as a nearest neighbor search or clustering, when applying DTW instead of a metric. Other lower-bound methods for DTW were introduced e.g. by [48, 32], also implemented in [35].

For a more detailed definition of DTW please see Paper B, or Paper C or [49], [20].

3.2 Time Series Clustering

Clustering time series is basically similar to clustering objects that are not time series. Given a set of time series, the challenge is to find a model that builds groups of most homogeneous time series. Finding the most suitable model for a set of data depends on the choice of a) the distance/similarity measure, b) the clustering algorithm, and c) the representation/preprocessing of a time series. Apart from the raw representation of a time series x , common practice is either to perform some preprocessing – for example normalization or smoothing –, or to represent a time series as set of derived features and statistics of x . If $x \in \mathbb{R}^{T \times M}$, then the derived set of k features is a vector $\Theta \in \mathbb{R}^k$. Section 3.3 elaborates how sliding algorithms can support the calculation of a feature vector Θ for all the segments of a time series x . The best choice of representation highly depends on the data and the use case, and typically some experimentation is required to meet the best decision for a dataset and use case.

Clustering Time Series Segments

For a time series x_t with observations ranging from 1 to T , a segment of x starting at index s with a length of m observations is defined as:

$$\{x\}_s^m := \{x_t | s \leq t \leq s + m - 1\} \quad \forall s, m > 0 \text{ and } s + m - 1 \leq T \quad (3.4)$$

Figure 3.2 illustrates some of the initial segments of x , all of the same length. Obviously two neighboured time series segments $\{x\}_s^m$ and $\{x\}_{s+1}^m$ are very similar. When

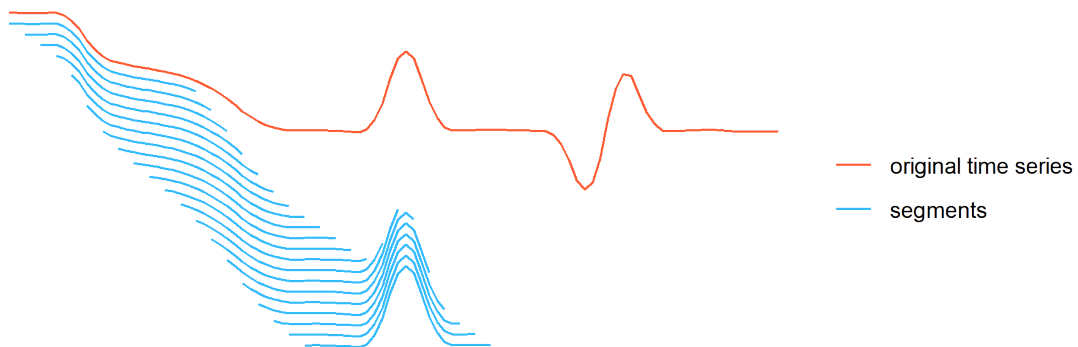


Figure 3.2: A time series x and some segments of x . The segments are plotted with a vertical offset.

attempting to cluster overlapping segments of a time series trajectory, the clustering task can lose its meaning. This is because the nearest neighbor of a segment is often a trivial neighbor that has a high overlap, starting either one time index earlier or later. [31] discuss this problem of meaningless time series segments clustering in detail.

Nevertheless, there is often a need to uncover structures and repeating patterns in a time series, so clustering time series segments can still be worthwhile and provide new insights into the data. In Paper D we developed an algorithm for clustering time series segments that circumvents this pitfall and guarantees that two overlapping neighbors can only end up in the same cluster, if both are connected to another segment, which has no overlap to any of the former two.

Apart from clustering the segments of a time series, the next section discusses other pattern recognition algorithms to uncover structures and repeating patterns in a time series.

3.3 Pattern Recognition and Sliding Algorithms

Given a long time series x , and a statistic θ to be computed for each of the segments $\{x\}_s^m$ of x . The naive brute force approach would be to first extract all of the segments and save these in a database Ω , and second to apply θ on all of the segments independently from each other.

In contrast, there are time series sliding algorithms, that apply computational steps on the segments of x incrementally. Depending on θ , there may be a function $f()$ that fulfills:

$$\theta_{s+1}^m = f(\theta_s^m, x_s, x_{s+m}), \quad (3.5)$$

and the calculation of θ_{s+1} via $f()$ is significantly faster than the brute force calculation, since there is no need to touch all of the observations of x in-between x_s and x_{s+m} . The initial value θ_1^m is to be calculated traditionally, independent from $f()$. The two observations x_s and x_{s+m} have a special role since $x_s \notin \{x\}_{s+1}^m$ and $x_{s+m} \notin \{x\}_s^m$, as Figure 3.3 illustrates by repeating the time series example and two of the segments of Figure 3.2, and emphasises which data is required to calculate the statistics θ_s and θ_{s+1} . This principle has also found application in [46]. The following formula 3.6 demonstrates step by step how to achieve the relation between two consecutive values of θ , for the

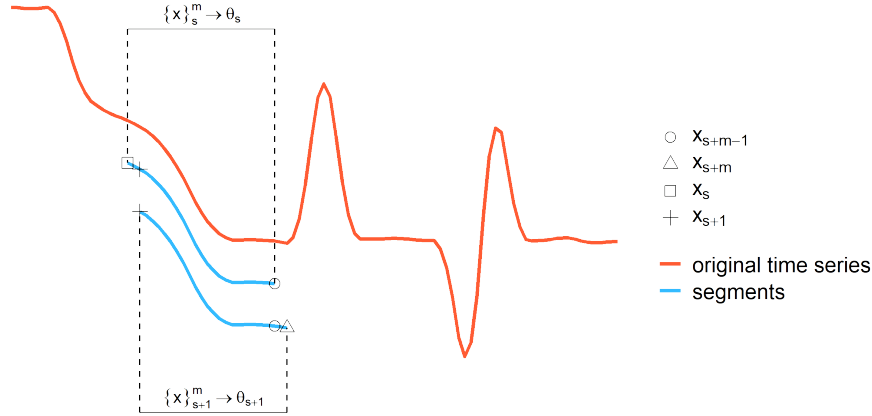


Figure 3.3: Incremental calculation of the statistic θ by recycling previous calculation results.

exemplary case of θ being the mean of a segment, μ :

$$\begin{aligned}
 \mu_{s+1}^m &= \frac{1}{m} \sum_{i=s+1}^{s+m} x_i \\
 &= \frac{1}{m} (x_{s+1} + \cdots + x_{s+m}) \\
 &= \frac{1}{m} (x_{s+1} + \cdots + x_{s+m}) + \frac{1}{m} (x_s - x_s) \\
 &= \frac{1}{m} (x_s + \cdots + x_{s+m-1}) + \frac{1}{m} (-x_s + x_{s+m}) \\
 &= \mu_s^m + \frac{1}{m} (x_{s+m} - x_s)
 \end{aligned} \tag{3.6}$$

So the runtime complexity of an update step of θ reduces from $O(m)$ to $O(1)$, and to calculate θ for all segments reduces from $O(T \times m)$ to $O(T)$.

Obviously it depends on the statistic θ if a function $f()$ exists, which fulfills $\theta_{s+1} = f(\theta_s, x_s, x_{s+m})$. Therefore this principle is not applicable in general for any statistic θ . However, the algorithms and methods presented in this thesis repeatedly apply this principle of incrementally updating statistics: Paper C and B for the update of the local cost matrix, and the update of the vector based DTW computation for new observations, in [35] for the update of the normalization of segments via the mean and standard deviation, and in Paper D again for the normalization of segments, and for updating the rotation matrix required to rotate a new segment to minimize its distance to a reference trajectory.

In summary, when faced with pattern recognition problems in time series mining, it can be advantageous to reformulate the algorithm so that the time series are analyzed in a sliding fashion. When time series segments are compared, often the distance calculation follows a preprocessing step of local normalization or calculation of statistics (see e.g.

[58], [7]). Depending on θ and the use case, a sliding algorithm can save computation time and the amount of the required storage compared to a brute force approach where all segments are stored in a database.

Chapter 4

Challenges and Approaches from Mobility Research

Mobility research deals with the questions of why and how people or goods are transported and what dynamics and consequences the transport behavior of a society has in a globalized world. Section 1 introduces the motivation of this thesis out of the perspective of mobility research and trip reconstruction, which is to develop efficient machine learning algorithms, that support data driven transport visibility systems. A transport visibility system is a digital platform or software application that provides real-time and end-to-end visibility into the movement and status of people, goods and shipments within a supply chain or transportation network. Efficient and accurate machine learning methods that applied in transport visibility systems could help to facilitate and evaluate a mobility shift towards sustainable transport means.

The following sections detail the main research questions relevant for this thesis out of the field of mobility research – especially for trip reconstruction and transport visibility – and what algorithms and methods this thesis proposes to approach these challenges.

4.1 Trip Reconstruction

In mobility research a transportation trip (or just trip) is the action of moving an object or a person from an origin to a destination. Trip reconstruction means to gain as much information about a recorded trip as possible, such as the trip purpose, the motivation for decisions along the trip, the set of used transport modes, the taken routes, and the duration of the mode-specific stages [8]. The data collected with travel-diaries can be used to develop methods for trip reconstruction.

The related field of transport visibility is the surveillance of a trip by tracking and monitoring the status and location of the moving object, person or vehicle from origin to destination. The challenge of these two fields is similar, to gain high level knowledge about events during the trip. The difference is, trip reconstruction usually is performed a posteriori by analysing historical data, and transport visibility analyses the data in real-time or near-real-time when the trip is actually recorded. The algorithms for these

two types of problems are also called offline and online methods. From a methodological stand point the methods to analyse the data to accomplish the challenge for trip reconstruction or transport visibility are very similar. The methods proposed in this thesis can be applied for either of these two problems.

Transport Mode Detection

Trips can be recorded with smartphones and special smartphone applications to record sensor data. With the help of this collected data, machine learning and data mining models are trained that help to gain the intended higher level information [43], [57]. An example for such a model is a transport mode detection model, which receives the collected sensor data as input and returns the most probable transport mode. Having such a model of high quality is key to perform smartphone supported and user-friendly mobility surveys to gain high quality insights about a population's mobility behaviour.

One challenge lies in the chicken-egg problem of the requirement of labelled data, to train a classifier model, which in turn is capable of labelling (without user interaction) new data, collected when the system is applied in the field. To resolve this problem, some manually labelled data needs to be collected in an initial phase. Once an initial model is ready, users can record trips and the model classifies the detected transport modes. If the system allows the user to adjust estimated labels provided by the model, the system could learn from this new user-provided ground truth data of the form: When did a trip start and end, when were transport modes changed, and what were the transport modes. However, labelling such data is a challenging task, because there is some room for interpretation in these questions. For example consider the following intermodal trip (an intermodal trip consists of a chain of various transport modes to get from an origin to the destination): a user is riding the metro and next takes the bus to reach their destination. Is there a segment of walking in-between the two segments of metro and bus? And when would be the correct point of time to split the segments between metro and walking, when the user exits the metro station, or when the user exits the metro car, or as soon as the user starts walking inside the metro towards the door to get off the metro?

Consequently it is a difficult task for users to provide valuable feedback to the system, and further it is challenging to incorporate user provided feedback in the training process of a model. This thesis approaches this challenge of systematically biased ground truth data on the one hand via an algorithm for improving a user-provided segmentation (see Paper A), and on the other hand via a data gathering initiative specifically designed to collect unimodal trips only (see Section 4.1). Further algorithms presented here are unsupervised methods trained on the unimodal dataset, and so predominantly free of erroneous ground truth data.

Trip Segmentation

As the example of an intermodal metro-bus ride in the previous section points out, the trip reconstruction not only consists of the transport detection but also of the problem

to find the best points in time to separate the trip into unimodal segments.

Paper A elaborates in detail that it is no trivial task for a user to set the point of time for changing the transport mode. And further, at what extend disturbances in the segmentation can cause problems in the feature extraction and actual training of a classifier model. The proposed algorithms SEG-IP and Segment-123 in Paper A adjust the user-stated segmentation by identifying changes in the variance of the accelerometer signal. The corrected segmentation results in more homogeneous segments. The experiments demonstrate that applying our proposed method helps a classification model to separate the modeled transport modes and in general to improve the classification performance.

Data Gathering

In the literature ([25], [39]) the data gathering for trip reconstruction problems is often performed with participants who are equipped with smartphones and special smartphone applications. A number of sensors are usually built into smartphones that could be of interest for training transport mode detection models. [25] However, sensor recording requires some battery capacity, so there is a trade-off between collecting as much data as possible to improve data quality and collecting as little data as necessary to reduce smartphone battery drain. This thesis puts special focus on algorithms for trip reconstruction designed for time series data from two different sensor readings: the GPS sensor and the accelerometer sensor. Both sensors are strongly represented in the literature of trip reconstruction ([25], [38], [39], [28], [52]). The GPS sensor delivers information in the spatial-temporal dimension about the movement trajectory of a device that carries the GPS sensor. On the other hand, the accelerometer sensor measures accelerations of the device in the 3 dimensions (x,y,z) independently of the device's location. Recording high frequency sensor data causes the battery of the smartphone to drain faster [25], and so the usability of a smartphone application – based on battery-intensive data collection – declines. The power consumption varies among the location sensors and motion sensors of a smartphone. Recording GPS data is very expensive, whereas the accelerometer sensor is rather inexpensive and still covers much information about human movements. Further, especially GPS sensors are affected by recording gaps. In urban canyons, inside buildings, in subways, or in special trains the signal is lost, and so the location information can hardly be rebuilt. Recording accelerometer data is unaffected by these two issues. Consequently a model to analyze trip data relying solely on accelerometer data and independent of GPS information is of advantage and can work as stand alone model or support a GPS-based system.

A special data gathering initiative helped to tackle the problem of erroneous ground truth data. On the basis of an already existing system by the AIT, the system enhances an Android smartphone application to collect sensor data with a smartphone during a transport trip is performed. Figure 4.1 depicts the user interface of the application. The user interface was designed intentionally as simple as possible, because experiments have shown that a simple usability helps participants to annotate the recorded trips accurately. Equipped with the smartphone application, the participants of the study

were advised to collect a number of unimodal trips. As soon as the trip was started, the participants were asked to state the respective transport mode, hence to annotate the ground truth label. At the end of the recording there was the possibility of adding some notes. Finally the trip was saved and uploaded to a server. It is important to remark, that the fact that the collected trips were unimodal, made it possible to identify transport mode-specific patterns.

While a GPS sensor records independently of user-interactions with the smartphone (as long as the GPS is activated), the accelerometer readings are sensible to the position in which the participant is carrying the phone, and whether the position is changed, or the phone is used for other purpose during the recording (e.g. texting messages, reading, etc.). [21] address this issue via recording trips with multiple smartphones equipped to the same participant in different positions. [25, 38, 39] apply a gravity component projection of the accelerometer signal, which we also applied in Paper B. A thorough analysis of the performance of transport mode identification models depending on various typical user interactions with the smartphone remains for future work. Developed on the



Figure 4.1: Layout of the smartphone application applied in the mobility data gathering initiative.

basis of this unimodal dataset this thesis proposes the Caterpillar algorithm (Paper B) that helps to identify 'vertically shifted and time warped matches of different lengths of hypothesis time series.' This way the Caterpillar algorithm can for example identify the typical pattern of a metro stop in accelerometer time series. And further, a trip reconstruction model could benefit from this information of a detected metro stop to segment the sensor time series in more homogeneous segments and finally to improve the quality of the trip reconstruction.

4.2 Recognition of Mobility Patterns

Mobility patterns can origin from any human, device or vehicle – equipped with a recording sensor – that performs a specific action or movement. Examples for such patterns and movements are

- the stop of a metro (see Figure 1 and 10, Paper B),
- turns of different vehicles, speeds and angles (see Figure 1, 10, 11, Paper D)
- a combination of the above, that symbolises for example a parking maneuver
- drinking a glass, walking, brushing teeth, writing a digit (see Figure 4, Paper C)

In the domain of recognising mobility patterns this thesis distinguishes between two challenges: The detection of patterns, and the retrieving of patterns.

- **Pattern retrieval:** Given a set of sensor readings and a query pattern, the task is to find similar occurrences of the query pattern in the set of sensor readings. The occurrences are also called fits or matches. These tasks are also called range query (or ϵ -query) and knn-query. The ϵ -query says: Given a set of time series X and a time series y , and a distance measure $d(.,.)$, find the set

$$\{x | d(y, x) < \epsilon \text{ and } x \in X\}. \quad (4.1)$$

The knn-query is similar: Given a set of time series X and a time series y , a distance measure $d(.,.)$ and a number of neighbors to be found k , find the set:

$$\{x | d(y, x) \leq d(y, z) \text{ where } x \in X^k, z \in X \setminus X^k, X^k \subseteq X, |X^k| = k\}. \quad (4.2)$$

- **Pattern detection:** Given a set of sensor readings, the task is to discover outstanding patterns or recurrent patterns, typically without the prior knowledge how a pattern might look, or how often it occurs, nor what it means. Detected patterns can be prototypical patterns for any context the sensor readings originate from (e.g. the detected u-turn in GPS trajectories of a dredger ship in Paper D). Methods from the domain of outlier detection or clustering are typically the methods of choice to address this problem.

Both problems, pattern retrieval and pattern detection depend on the length of the desired pattern y . And if y is significantly shorter than the time series in X , than sliding algorithms (see Section 3.3) can help to increase the efficiency to solve these problems.

This thesis faces both problems and proposes distance measures and algorithms to address these problems.

Section 4.1 already introduced trip reconstruction and mentions how the Caterpillar algorithm from Paper B can support the reconstruction of a trip. The results section of this paper demonstrates that the Caterpillar algorithm is for example capable of

retrieving query patterns of accelerometer readings of a metro, that decelerates before a stop, then pauses, and accelerates again to continue the ride. The applied distance measure is a marriage of the DTW distance measure and the MDL principle. This combination results in a method that decides without the need of a user-given threshold whether a segment of a signal is similar to the query pattern or not. Further, the Caterpillar can retrieve similar patterns, even though they are of different time extend. So, coming back to the metro example, metro stops were identified to be similar matches of the query metro stop, even though the metro waited in some stops half or twice as long as it waited in the query pattern.

Paper D deals with both challenges, detecting and retrieving patterns in GPS trajectories. The idea of this paper is to measure similarities and recognise patterns invariant of any rotation, location. This means, that a simple movement as a left turn of a driving car is similar to another left turn, no matter where located and in which cardinal direction the car is heading during the two left turns. The same applies to more complex movements. Based on a novel rotation-invariant distance metric the algorithm S3KR is proposed to scan a GPS trajectory for retrieving similar segments of a query pattern, as for example retrieving all left turns in a GPS trajectory of some hours. On the other hand, S3KR is applied in a second proposed algorithm, TSS, to cluster the segments of a trajectory and to detect recurrent, prototypical patterns of movements in a GPS trajectory. A case study in Paper D presents a GPS trajectory of a dredger ship. At first glance, it is difficult to identify a purpose in the ship's movements, since the GPS trajectory looks like a blob of GPS positions that seem to origin from a ship moving in a disoriented zick-zack pattern. Applying the algorithm TSS reveals the most frequent movement patterns of the ship: u-turns. This way it was easy to interpret, that the ship was circling around a certain position in the sea. Another case study in that paper applies the proposed methods to compare prototypical movement patterns of trains an light-rail GPS trajectories. The comparison shows that light-rail trains operate on a rail network that enables the train-cars to make turns of smaller radius.

4.3 Runtime Efficient Algorithms

In general, runtime efficient algorithms are important because they can solve complex problems quickly and allocate computational resources – such as CPU time and memory – more efficiently. As the size of data sets and computational problems continue to increase, the importance of efficient algorithms becomes obvious, when formerly intractable or too cost intensive problems become solvable due to more efficient programming. Further, efficient programming can also reduce the amount of energy required to analyse data or solve complex problems. The European Green Deal [19] points out the importance of efficient algorithm development in the realm of Artificial Intelligence: "The future contribution of AI to GHG emissions will depend on the energy efficiency of data centres and their operation with renewable energies. Furthermore, it is crucial to make energy and resource efficiency a dedicated development goal in the AI innovation process."

As in other domains, also in mobility research runtime efficiency is crucial. On the one hand, the proliferation of mobile devices and sensors results in an inundation of machine-generated trajectory data about moving people, vehicles, vessels and other objects. On the other hand, multiple use cases in mobility research require efficient data processing for real time or near-real time response, such as navigation systems, recommendation systems [4] or ticketing apps for public transport that require to detect the state of the user. Further, the user experience and consequently the user acceptance benefits from instantaneous responses of smartphone application. This is a critical factor, when smartphone applications – that require data processing by the help of machine learning algorithms – are designed to ease the transition towards environmentally friendly transport means. For this reason, this thesis puts a special emphasis on the runtime efficiency of novel algorithms, especially by designing the proposed algorithms according to two principles: Firstly, by recognizing which calculation steps are necessary to return exact results, and skip those that are not necessary, and secondly by recycling already calculated results.

Paper C proposes the software package IncDTW [35] to calculate the DTW distance measure incrementally. Considering a system where data is updated every second and predictions based on distance measures need to be calculated in real time, the incremental update of the DTW distance measure helps to facilitate these real time predictions. Paper C demonstrates this via an exemplary classifier model that processes accelerometer sensor data and also updates the prediction of the recognized activity every second. Further, the principle of incremental calculation of the DTW distance measure finds application in Paper B.

Paper B addresses the problem of searching long time series to retrieve query patterns. The proposed Caterpillar algorithm decides whether a segment is a representation of the query pattern by applying the MDL principle. The algorithm scans time series by combining the incremental calculation from Paper C and the MDL principle. The Caterpillar algorithm is capable to retrieve the best matches of query patterns that represent e.g. a metro that stops-pauses-accelerates again.

Also the algorithm S3KR proposed in Paper D is developed for scanning time series efficiently. Especially, S3KR is designed to scan GPS trajectories and discover similar trajectory segments independent of their position and orientation. The key milestones for S3KR to improve runtime efficiency are: a) the introduction of a novel rotation invariant distance metric that allows to decouple the rotation from the standardisation, b) the incremental update of statistics required for standardization (as discussed in Section 3.3), and c) the early abandoning of unnecessary calculation steps, also possible due to a) and b).

Chapter 5

Contributions, Conclusion and Perspectives

5.1 Contributions

The research work for this thesis has been motivated out of two domains, the domain of transport visibility as subdomain of mobility research, and the more general domain of time series mining.

From the point of view of the transport visibility domain we contributed by analysing the impact of the segmentation bias – which is a systematic bias in the ground truth data caused by erroneous user input – and developed the algorithms SEG-IP and Segment-123 that improve a user-provided segmentation of a trip record to find more homogeneous data segments. Our methods improve the quality of the training data, and ultimately result in the presentation of a semi-supervised transport mode classifier, which is more robust against biased user input, Paper A. Further we developed the Caterpillar algorithm, which can also help to split a trip into segments by detecting patterns of interest, Paper B. The Caterpillar algorithm is capable of identifying transport mode specific patterns in accelerometer sensor readings of smartphones. The algorithm is not limited to being applied to this type of data, but can be applied to any type of time-series data, in particular to analyze patterns that may vary in extent over time. Moreover, in Paper D we developed methods to analyse GPS trajectories for detecting and retrieving patterns invariantly of any rotation. This can help to analyse movement patterns that are specific for transport modes, moving objects or activities. Overall we demonstrated in the respective publications that our proposed methods help to get a better understanding of sensor readings, either accelerometer or GPS, related to recorded trips or trajectories. Our proposed pattern recognition methods help to address both problems out of transport visibility, trip segmentation and transport mode detection.

In the realm of time series analysis, our contribution manifests through the methodological developments to analyse mobility data. The Caterpillar algorithm in Paper B combines the time series distance measure DTW and the well known principle of MDL. This marriage facilitates on the one hand the parameter-free retrieval of query patterns

in time series. And on the other hand, by exploiting the nature of the DTW algorithm, the patterns retrieved can vary in temporal extent. During the development of the Caterpillar algorithm, the need for a software toolbox to incrementally calculate DTW distance arose. Paper C presents the R package IncDTW that fills this gap. IncDTW is mainly about the incremental DTW calculation, and supporting functionalities to give a user the possibility to write their own algorithm that applies incremental DTW computation. Further, IncDTW is the first package on CRAN that offers the vector-based implementation of the DTW algorithm – for multivariate time series and/or time series of various lengths – which considerably decreases the computation time. Our R package is also mentioned in the task view of time series ¹, which is a collection of selected R packages for various time series analysis tasks. For analysing GPS trajectories Paper D presents our methods to perform rotation invariant mining of patterns in time series. The proposed methods are (1) a novel rotation-scaling-shifting invariant distance metric, (2) the sliding algorithm S3KR that efficiently tackles the kNN-query or range-query, and (3) an the algorithm TSS for clustering the segments of a time series – by respecting the overlap of segments – to discover groups of segments in the time series and further to detect prototypical patterns in a time series.

During the development of the contributed methods in this thesis we always put a special emphasis on computational efficiency of our proposed algorithms. This supports a better usability, a broader applicability and reduced energy consumption. Our methods achieve these efficiencies by taking advantage of two principles: allocate as little as necessary and limit the computational steps to those that are inevitable. The first principle found application via the vector based implementation of DTW in Paper B and C. We realized the second principle by remodelling mathematical foundations of our methods to finally either recycle computational results in sliding algorithms, or early abandon computations (Paper B, C and D).

5.2 Conclusion and Discussion

This thesis is a composition of novel data mining algorithms for time series, specifically but not exclusively out of the domain of mobility research.

We tackled the problem of biased user input and time series segmentation, on the one hand via the development of the novel algorithms Seg-IP and Segment123 to revise the user input, similar to [30, 26] who apply change detection in a signal for segmentation. And on the other hand, via the design of a special data gathering initiative, supported by a smartphone application.

Further we focused on the problem of detecting patterns in time series that give more details about the segmentation and cause of the underlying data generation process. To that end we put significant effort in one basic element in the data mining process, measuring the similarity of time series and segments of time series. We focused mainly on two distance measures that are most relevant and widely applied in the literature [51, 46, 61], the Euclidean metric and Dynamic Time Warping.

¹<https://cran.r-project.org>

Motivated by the use case to detect transport mode specific patterns that can vary in the temporal extend, and inspired by the pattern detection algorithm UCR Suite [46], which applies DTW, we developed the Caterpillar algorithm, which marries DTW and MDL. The Caterpillar algorithm recognizes if a segment of varying length in a time series is a close-enough representation of a query pattern or not. And due to the combination with the MDL principle, the Caterpillar makes this decision parameter-free, while setting such a threshold parameter would be a difficult task for a user. We also demonstrated to outperform the related approaches from literature for our use case. To operate most efficiently, the foundational work for the Caterpillar algorithm is the incremental computation of DTW, which we implemented and published as R package [35], Paper C. Some of the most popular approaches in the literature to increase the efficiency of the DTW computation are lower bounding and early abandoning [33, 48], the vector-based implementation of the DTW core algorithm [5], and the incremental computation of DTW [45, 11, 40]. We implemented all of these in [35] and also introduced the vector-based incremental calculation of DTW.

There are two established approaches for analysing accelerometer data independently from rotation and position of the recording device. The first is to calculate the absolute power of the signal, which is the 2-norm [42, 56], and the second is to use the projection of the 3-dimensional signal into the 2 dimensions of horizontal and vertical component [25, 38, 39], Paper B and A. We could not find equivalent methods in the literature for the analysis of GPS patterns, which suffice our requirements of analysing efficiently and robust against noise. For this reason we developed the rotation invariant distance measure inspired by Procrustes analysis, the sliding algorithm S3KR, and the time series segment clustering algorithm TSS. The most similar approaches we found in the literature addressed this either via an projection of the GPS trajectory into the space of arc length and angle with the Euclidean distance [18] or DTW as distance measure [53], or via the transformation towards a series of distances from its own gravity center [59]. We compared our proposed distance method with the similarity measures presented in these papers and outperformed them in several experiments in accuracy and efficiency. Further the similarity measures from literature were not optimized for application in a sliding algorithm as we proposed, to most efficiently mine big amounts of GPS trajectories.

During recent years deep learning, or deep neural networks (DNN) (e.g.: Convolutional NN, Recurrent NN, Long-Short-Term Memory, Generative Adversarial Network, etc.) seem to have ushered in a new era of machine learning, while this thesis relies on the development of algorithms that rely on traditional machine learning and data mining methods compared to DNN. DNN excel in various data mining and machine learning tasks, especially applied on structured data as images or time series. However, these methods also share some disadvantages. First, DNNs are not the first choice for every use case and data set, since the training of these models with hundreds to thousands of parameters requires a big amount of data, dependent on the complexity of the problem and the number of parameters. Next, due to the huge amount of parameters, DNNs can be difficult to interpret, which is often appreciated when there is the need to communicate the results to stakeholders, or to develop, monitor and debug a system that applies

a DNN. Further, for the use case of edge computing – deployment of an application, based on a machine learning model, directly on a mobile device – it is necessary that the model is manageable in size, and efficient in prediction. Moreover, Occam’s razor would suggest to use the simple model instead of the more complicated in case both approaches perform equally well. Finally, in a use case where the application of a DNN is highly advantageous, the methods presented in this thesis can also be applied in an ensemble model, where for example detected distances to a given query pattern serve as input to a DNN that uses other data as well. For these reasons, there will likely be a long-term need for traditional machine learning and data mining models, along with the need for DNNs.

We accompanied the research work for this thesis with comprehensive experimenting and a thorough comparison with the related work in the respective publications (Paper A to D). For the use case of transport visibility and time series mining our publications demonstrate that our developed methods contribute to the state of the art.

This thesis concludes that the proposed methods can help other researchers and users to analyse time series data most efficiently and achieve a better understanding about a dataset, which can serve as stand alone result, or as input for further analysis methods.

5.3 Outlook

The domain of transport visibility and time series mining consist of a plethora of open research questions. The methods presented in this thesis address some of these problems. Future work in applied research based on this thesis could deal with the integration of our proposed algorithms in applications to support transport visibility. Our algorithms for pattern detection can be integrated in ensemble models in combination with state of the art machine learning models for supervised/unsupervised learning problems. Furthermore, a thorough analysis of the performance of transport mode identification models depending on typical user interactions with the recording smartphone would be necessary to comprehensively evaluate a system applied for trip reconstruction.

Future work in the methodological perspective could deal with the integration of DTW in the rotation-invariant distance measure δ and the sliding algorithm S3KR (Paper D). The advantageous non-linear assignment of DTW could help to improve the resilience of S3KR against outliers and varying speeds in similar patterns. At the same time, designing an algorithm that combines S3KR with DTW would inherit the challenge of keeping it runtime efficient, to remain the applicability. Finally, the implementation of the algorithm TSS (Paper D) for DTW in the R package IncDTW is planned.

Bibliography

- [1] *Partitioning Around Medoids (Program PAM)*, chapter 2, pages 68–125. John Wiley Sons, Ltd, 1990.
- [2] Neamah Al-Naffakh. A comprehensive evaluation of feature selection for gait recognition using smartwatches. 2017.
- [3] Ghazi Al-Naymat, Sanjay Chawla, and Javid Taheri. Sparsedtw: A novel approach to speed up dynamic time warping. *arXiv preprint arXiv:1201.2969*, 2012.
- [4] Olga Artemenko, Volodymyr Pasichnyk, and Nataliia Kunanets. Using mobile location-based recommender systems for providing real time recommendations for social distancing urban route planning. In *2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT)*, volume 2, pages 305–308, 2020.
- [5] Philipp Boersch-Supan. rucrdtw: Fast time series subsequence search in r. *The Journal of Open Source Software*, 1:1–2, 2016.
- [6] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. Springer, 2002.
- [7] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.
- [8] Eui-Hwan Chung and Amer Shalaby. A trip reconstruction tool for gps-based personal travel surveys. *Transportation Planning and Technology*, 28(5):381–401, 2005.
- [9] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd VLDB conference, Athens, Greece*, pages 426–435. Citeseer, 1997.
- [10] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, 2004.
- [11] Simon Dixon. An on-line time warping algorithm for tracking musical performances. In *IJCAI*, pages 1727–1728, 2005.

-
- [12] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.
- [13] Oya Ekin, Peter L Hammer, Alexander Kogan, and Pawel Winter. Distance-based classification methods. *INFOR: Information Systems and Operational Research*, 37(3):337–352, 1999.
- [14] Charles Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th international conference on Machine Learning (ICML-03)*, pages 147–153, 2003.
- [15] Adrian Ellison, Richard Ellison, Dean Rance, Chris Standen, Chris Rissel, and Melanie Crane. A Web-Based Diary and Companion Smartphone app for Travel/Activity Surveys. *Transportation Research Procedia*, 11:297–310, 2015.
- [16] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.
- [17] Constanze Fetting. The european green deal. *ESDN Report, December*, 2020.
- [18] Udo Feuerhake. Recognition of repetitive movement patterns—the case of football analysis. *ISPRS International Journal of Geo-Information*, 5(11):208, 2016.
- [19] Peter Gailhofer, Anke Herold, Jan Peter Schemmel, Cara-Sophie Scherf, Cristina Urrutia de Stebelski, Andreas R Köhler, and Sibylle Braungardt. *The role of artificial intelligence in the European Green Deal*. European Parliament Luxembourg, Belgium, 2021.
- [20] Toni Giorgino. Computing and visualizing dynamic time warping alignments in r: the dtw package. *Journal of statistical Software*, 31:1–24, 2009.
- [21] Hristijan Gjoreski, Mathias Ciliberto, Lin Wang, Francisco Javier Ordonez Morales, Sami Mekki, Stefan Valentin, and Daniel Roggen. The university of sussex-huawei locomotion and transportation dataset for multimodal analytics with mobile devices. *IEEE Access*, 6:42592–42604, 2018.
- [22] Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.
- [23] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.
- [24] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

- [25] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, SenSys '13*, pages 13:1–13:14, New York, NY, USA, 2013. ACM.
- [26] Johan Himberg, Kalle Korpiaho, Heikki Mannila, Johanna Tikanmaki, and Hannu TT Toivonen. Time series segmentation for context recognition in mobile devices. In *Proceedings 2001 IEEE international conference on data mining*, pages 203–210. IEEE, 2001.
- [27] IEA. World energy outlook 2022, 2022.
- [28] Sahak Kaghyan and Hakob Sarukhanyan. Activity recognition using K-nearest neighbor algorithm on smartphone with Tri-axial accelerometer. *International Journal of Informatics Models and Analysis*, 1:146–156, 2012.
- [29] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.
- [30] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Proceedings 2001 IEEE international conference on data mining*, pages 289–296. IEEE, 2001.
- [31] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *KAIS*, 8(2):154–177, 2005.
- [32] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.
- [33] Eamonn Keogh, Li Wei, Xiaopeng Xi, Sang-Hee Lee, and Michail Vlachos. Lb_keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *Proceedings of the 32nd international conference on Very large data bases*, pages 882–893. VLDB Endowment, 2006.
- [34] Marzena Kryszkiewicz and Piotr Lasek. Ti-dbscan: Clustering with dbscan by means of the triangle inequality. In *Rough Sets and Current Trends in Computing: 7th International Conference, RSCTC 2010, Warsaw, Poland, June 28-30, 2010. Proceedings 7*, pages 60–69. Springer, 2010.
- [35] Maximilian Leodolter. R-package for incremental dynamic time warping, July 2017. <https://cran.r-project.org/web/packages/IncDTW/index.html>.
- [36] Maximilian Leodolter, Claudia Plant, and Norbert Brändle. Theory and applications for the r package incdtw. <https://cran.r-project.org/web/packages/IncDTW/index.html>.

- [37] Stef Lhermitte, Jan Verbesselt, Willem W Verstraeten, and Pol Coppin. A comparison of time series similarity measures for classification and change detection of ecosystem dynamics. *Remote sensing of environment*, 115(12):3129–3152, 2011.
- [38] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, SenSys '10*, pages 71–84, New York, NY, USA, 2010. ACM.
- [39] David Mizell. Using gravity to estimate accelerometer orientation. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers, ISWC '03*, pages 252–, Washington, DC, USA, 2003. IEEE Computer Society.
- [40] Akihiro Mori, Seiichi Uchida, Ryo Kurazume, Rin-ichiro Taniguchi, Tsutomu Hasegawa, and Hiroaki Sakoe. Early recognition and prediction of gestures. In *18th International conference on pattern recognition (ICPR'06)*, volume 3, pages 560–563. IEEE, 2006.
- [41] Rossana Muscillo, Silvia Conforto, Maurizio Schmid, Paolo Caselli, and Tommaso D'Alessio. Classification of motor activities through derivative dynamic time warping applied on accelerometer data. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4930–4933. IEEE, 2007.
- [42] Philippe Nitsche, Peter Widhalm, Simon Breuss, Norbert Brändle, and Peter Maurer. Supporting large-scale travel surveys with smartphones - A practical approach. *Transportation Research Part C: Emerging Technologies*, 43:212–221, 2014.
- [43] Philippe Nitsche, Peter Widhalm, Simon Breuss, Norbert Brändle, and Peter Maurer. Supporting large-scale travel surveys with smartphones—A practical approach. *Transportation Research Part C: Emerging Technologies*, 43:212–221, 2014.
- [44] Yiwei Pan, Zhibin Pan, Yikun Wang, and Wei Wang. A new fast search algorithm for exact k-nearest neighbors based on optimal triangle-inequality-based check strategy. *Knowledge-Based Systems*, 189:105088, 2020.
- [45] Lawrence Rabiner, A Rosenberg, and S Levinson. Considerations in dynamic time warping algorithms for discrete word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(6):575–582, 1978.
- [46] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *18th ACM SIGKDD*, pages 262–270, 2012.
- [47] Chotirat Ann Ralanamahatana, Jessica Lin, Dimitrios Gunopulos, Eamonn Keogh, Michail Vlachos, and Gautam Das. Mining time series data. *Data mining and knowledge discovery handbook*, pages 1069–1103, 2005.

-
- [48] Toni M Rath and R Manmatha. Lower-bounding of dynamic time warping distances for multivariate time series. *University of Massachusetts Amherst Technical Report MM*, 40:1–4, 2002.
- [49] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans Acoust Speech Signal Process*, 26(1):43–49, 1978.
- [50] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.
- [51] Patrick Schäfer. Scalable time series classification. *Data Mining and Knowledge Discovery*, 30(5):1273–1298, 2016.
- [52] Pekka Siirtola, Perttu Laurinen, Eija Haapalainen, Juha Röning, and Hannu Kinnunen. Clustering-based activity classification with a wrist-worn accelerometer using basic features. *2009 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2009 - Proceedings*, pages 95–100, 2009.
- [53] Michail Vlachos, Dimitrios Gunopulos, and Gautam Das. Rotation invariant distance measures for trajectories. In *Proceedings of the 10th ACM SIGKDD*, pages 707–712. ACM, 2004.
- [54] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26:275–309, 2013.
- [55] Xueyi Wang. A fast exact k-nearest neighbors algorithm for high dimensional search using k-means clustering and triangle inequality. In *The 2011 international joint conference on neural networks*, pages 1293–1299. IEEE, 2011.
- [56] Peter Widhalm, Philippe Nitsche, and Norbert Brändle. Transport mode detection with realistic smartphone sensor data. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 573–576. IEEE, 2012.
- [57] Peter Widhalm, Yingxiang Yang, Michael Ulm, Shounak Athavale, and Marta C. González. Discovering urban activity patterns in cell phone data. *Transportation*, 42(4):597–623, 2015.
- [58] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1317–1322. Ieee, 2016.
- [59] Dengsheng Zhang and Guojun Lu. Review of shape representation and description techniques. *Pattern recognition*, 37(1):1–19, 2004.

-
- [60] Yu Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.
 - [61] Yu Zheng. Trajectory data mining: An overview. *ACM Transactions on Intelligent Systems and Technology*, 6:1–41, 05 2015.
 - [62] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.
 - [63] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.

Publications

Paper A

Semi-supervised Segmentation of Accelerometer Time Series for Transport Mode Classification

License Information

©2017 IEEE. Reprinted, with permission, from Leodolter et.al., Semi-supervised Segmentation of Accelerometer Time Series for Transport Mode Classification, IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2017.

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Vienna's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

Semi-supervised Segmentation of Accelerometer Time Series for Transport Mode Classification

Maximilian Leodolter*, Peter Widhalm*, Claudia Plant†, and Norbert Brändle*

*Center for Mobility Systems, AIT Austrian Institute of Technology, Vienna, Austria

Email: maximilian.leodolter@ait.ac.at

†Faculty of Computer Science, University of Vienna, Vienna, Austria

Abstract—Collecting ground truth data with smart phone applications is as difficult as important for training classification models predicting transport modes of people. Errors of respondent input with respect to trip length and transport mode segmenting introduce a systematic bias in the classification model. We propose a semi-supervised framework adjusting user-given input to process user-collected accelerometer time series data. Our contributions are (1) an evaluation of the impact of segmentation bias, (2) a novel algorithm to find more homogeneous segments and (3) a robust incrementally trained classifier model based on clustering employing Dynamic Time Warping as similarity measure. We apply the proposed method on synthetic and real-world accelerometer trip data of 800 labeled trips consisting of 2000 user-given segments and 400 hours travel time and test it against a baseline classifier relying completely on user-feedback. The results prove that our method learns clusters revised from noise and increases the classifier’s accuracy for real-world and synthetic data by up to 17%.

Keywords— Transport Mode Detection, Accelerometer, Segmentation, Clustering, Dynamic Time Warping

I. INTRODUCTION

In mobility research traditional pen and paper surveys have partially been replaced in favor of smartphone-based systems reconstructing multimodal trips of people, which have the potential to collect more accurate and longer-term mobility data with much less effort for the respondent [1], [2]. Such systems usually incorporate data-driven classification models to distinguish different travel modes such as walking or taking the bus. The classification is based on features extracted from smartphone sensor data, most notably GPS locations or accelerometer readings. Data-driven classification models require ground truth data for training and for evaluation of classification accuracy. But in general, obtaining ground truth data by any form of manual user interaction is challenging [3]. When applying a post-trip interactive graphical interface such as in [2], [4], our experience shows that there are two different kinds of erroneous user-input. First, users sometimes assign a wrong label to a segment. In the literature this kind of error is known as label noise and it has been shown that it can have a significant impact on the performance of classifiers [5], [6]. Second, it turned out that users cannot be expected to specify the exact times of transition between the trip segments. In particular, when trip data are recorded at high sampling rates and users are asked to indicate the transition points based on the recorded GPS track displayed on a map, it is practically

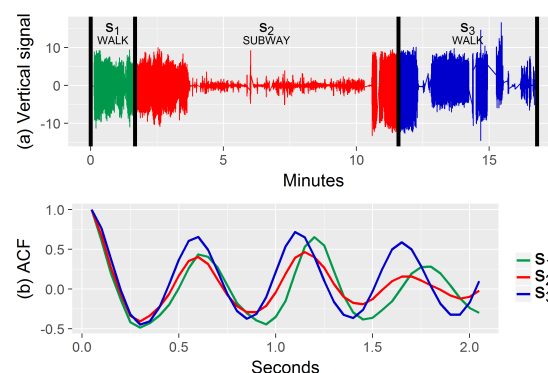


Figure 1: Impact of segmentation bias on the ACF. (a) shows the vertical component of the accelerometer signal of a trip. The user defined the transition points (vertical black lines) and the labels. (b) shows the similar ACF of the three segments.

impossible to obtain a precise segmentation. The reasons are that a) the higher the sampling rate the more difficult it is for the user to provide an exact segmentation of the data stream, b) the relationship between position on the map and points in time is ambiguous, and c) inaccurate positions on the map originating from measurement errors and interpolation directly translate to inaccurate segmentation. Here, we define this kind of error as segmentation bias. Existing literature on transport mode recognition frequently involves processing 3D-accelerometer data ([7], [8], [9]). The *vertical* component is extracted by aligning the 3-dimensional signal with an estimated Earth gravity vector [10]. The ACF of this signal or its Fourier Transform – the Spectral Density – is often used as one of the most relevant features (e.g. [7], [11]–[13]). Figure 1 illustrates (a) the vertical component of a trip’s accelerometer signal, and (b) the ACFs of three trip segments, defined and labeled by the user. The vertical lines in (a) depict the user-set transition points between segments. The two walk segments s_1 and s_3 show patterns of high variance, similar to the beginning and ending of the subway segment s_2 , while the middle part of s_2 has a much lower signal variance. However, the feature vectors, i.e. the ACFs, can hardly be distinguished. Considering that the user-given segmentation is probably inaccurate, this example motivates our hypothesis, that segmentation bias in raw user-supplied ground-truth data can severely impact classification accuracy.

In this work, we therefore evaluate the impact of segmentation bias on classification performance and propose a method to correct the user-supplied segmentation. Similar to [14] and [15] the proposed approach identifies changes in the signal’s variance to increase the homogeneity within each trip segment. Further we propose a robust iteratively-incrementally learned classifier to learn revised representatives per label. Finally we demonstrate the improvement achieved by our method with both synthetic and real-world accelerometer data recorded in Vienna, Austria. While the reported experiments use the ACF of the signal as feature, it can be expected, that the effects of segmentation bias demonstrated for the ACF also apply to all other features based on transformations of the signal to the frequency domain.

The remainder of this paper is organized as follows: Section II introduces the segmentation algorithm, which the classifier learning algorithm in Section III incorporates. Section IV details the evaluation and discusses the results. Finally we draw conclusions and point out future research in Section V.

II. SEMI-SUPERVISED SEGMENTATION

In the remaining paper a user-recorded trip is considered as a time series $t \in \mathbf{R}^N$ that can be separated in the time domain into segments s , where each s has a label l (which represents the transport mode). A point in time when one segment ends and the next segment begins is called a breakpoint b . We call a breakpoint *biased*, if it is different from it’s true value. Likewise, a segment s is *biased*, if at least one of the two enclosing breakpoints are biased.

Figure 1 motivates the approach to identify more homogeneous subsegments σ within a user-given segment. The proposed method identifies additional breakpoints of a signal by finding points in time when the signal’s variance changes significantly. The fitness of a breakpoint candidate b^c to separate a trip $t \in \mathbf{R}^N$ into the left and right subsegments σ_l and σ_r is defined as:

$$\text{fitness}(t, b^c) := g(t[1 : b^c], t[b^c + 1 : N])$$

$$g(\sigma_l, \sigma_r) := \max\left(\frac{V(\sigma_l)^2}{V(\sigma_r)}, \frac{V(\sigma_r)^2}{V(\sigma_l)}\right)$$

where V is the variance. Evaluating $\text{fitness}(t, b^c)$ for different b^c along t returns a time series of fitness values, of which local maxima indicate points in time where the variance of t changes significantly. This fitness function is used in the here proposed segmentation algorithm SEG-IP (Segmentation Incorporating Priors, see Figure 1), that incorporates user-given breakpoints, and looks for further breakpoints to finally form a set of subsegments where each is more homogeneous than the segment it origins from. B_t is the set of all breakpoints $\{b_i\}_{i=1\dots}$ of trip t and $\Sigma(t, B_t)$ is the set of subsegments $\{\sigma_i\}_{i=1\dots}$ of t formed by the enclosing breakpoints in B_t . Even though the given breakpoints are probably biased, they serve as borders defining restricted ranges which are searched for subsegments. Figure 2 depicts the fitness functions for different displacements of the two user-defined breakpoints

Algorithm 1 The algorithm SEG-IP (SEGmentation Incorporating Prior user-given breakpoints) achieves homogeneous subsegments of a trip

```

1: procedure SEG-IP( $t, B_t$ )
2:    $B_t^* \leftarrow B_t$  ▷ set of new breakpoints
3:    $F \leftarrow \text{matrix}(N, |B_t|)$ 
4:   for  $j \in 1 : |B_t|$  do
5:     for  $i \in b_{j-1} : b_{j+1}$  do
6:        $F[i, j] \leftarrow \text{fitness}(t[b_{j-1} : b_{j+1}], i)$ 
7:     end for
8:     add local maxima of  $F[:, j]$  to  $B_t^*$ 
9:   end for
10:  get and return  $\Sigma(t, B_t^*)$ 
11: end procedure

```

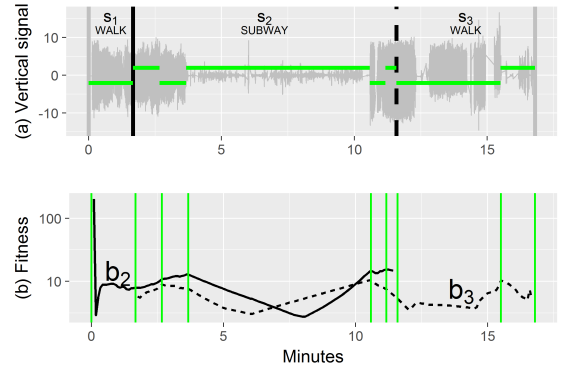


Figure 2: (a) shows the same signal as Figure 1a. The black vertical breakpoints (solid and dashed) in (a) relate to the fitness functions plotted in (b), solid and dashed respectively. The vertical green lines in (b) indicate the local maxima of the fitness functions to separate the segments into subsegments. The green horizontal lines in (a) indicate the new subsegments.

within the range defined by the previous and next breakpoint. The green vertical lines in Figure 2b indicate the local maxima, which will be used as corrected breakpoints. The corrected breakpoints enclose the new subsegments represented by green horizontal lines in Figure 2a. SEG-IP results in a new separation of a trip $\Sigma(t, B_t^*)$ and enables extracting features from these subsegments, instead of using biased user-defined segments. The example in Figure 1 is continued in Figure 3 showing (a) the new segmentation achieved by applying the algorithm SEG-IP and (b) 3 auto correlation functions:

- The red highlighted part of the signal in (a) represents the subway segment s_2 corresponding to the user-given segmentation, with the corresponding ACF in (b), also in red. A repetition from Figure 1.
- σ_2 (purple) is one of the subsegments that cause the bias of the ACF of s_2 .
- σ_4 (cyan) is the longest subsegment (and hence the most important representative) of s_2 . The importance of a subsegment in terms of influence onto the prediction model will be discussed in Section III.

This exemplary trip (depicted in the Figures 1, 2 3) and the comparison of the ACF for the noticeable segments and subsegments demonstrated that a biased segmentation – possibly user-given – can cause features extracted from these segments to be biased as well.

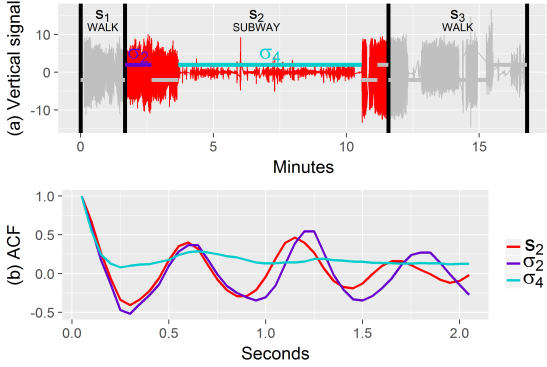


Figure 3: How segmentation bias can affect the ACF. (a) shows the same signal as Figure 1a with a focus on the second segment. (b) depicts the ACF of S_2 and the new subsegments σ_2 and σ_4 .

III. SEGMENT123-CLASSIFIER

This section describes a classification model based on a nearest medoid classifier (NMC) that we use in our experiments in Section IV on a synthetic dataset and a real-world dataset to evaluate the segmentation bias. The results as the evolution of the results (the cluster medoids) of a NMC are easy to interpret which is a key advantage over other classifiers (e.g. Neural Network, SVM, Logistic Regression etc.) if ground truth data is biased and so the interpretation of the results needs to reach beyond a binary true-false discussion (see Figures 5 and 7). The distance measure used for clustering and classification is Dynamic Time Warping.

A. Why Dynamic Time Warping?

Dynamic time warping (DTW) [16] is an algorithm for estimating similarity between two time series which possibly vary in speed. By allowing the sequences to be warped (compression and/or strain in time) similarities of shape independent of non-linear variations in time can be found. Section II made clear that such prudent walking signals mixed up with non-walking signals lead to a bias towards a walking pattern. To prevent such a mixup we apply DTW as similarity measure to facilitate differentiating patterns of different labels and joining similar patterns that are shifted in time within the clustering algorithm. DTW returns a small distance value for ACF that are slightly shifted, however have the same pattern. Especially for patterns of walking segments this is the case since the walking speed and step width differs from person to person and day to day. In order to demonstrate that we need the warping property in our application, consider the ACFs of two walk signals in Figure 4 which are very similar but slightly vary in speed. The Euclidean Distance is not able to recognize the similarity between those two time series. The distance between them is in the same order of magnitude as the distance between each of them to the zero line, which is a constant time series. In contrast, DTW transforms the signals such that the similarity of the basic pattern becomes obvious. Matching our intuition, the mutual distance between the two walk signals is very small compared

to their distance to the zero-line. The calculated similarity measures for the ACFs depicted in Figure 4 and their relations to the 0-line as comparison are $\frac{DTW(ACF_a, ACF_b)}{DTW(ACF_x, 0)} \approx \frac{7}{26}$ and $\frac{d_{euclid}(ACF_a, ACF_b)}{d_{euclid}(ACF_x, 0)} \approx \frac{21}{22}$, where $x \in \{a, b\}$. Consequently DTW is much more appropriate to assign small dissimilarity values for similar walking patterns than the Euclidean distance.

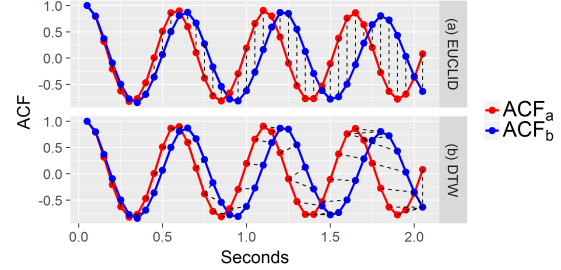


Figure 4: Why the ACF is much more appropriate than the simple Euclidean distance measure: (a) shows two ACF and the 1-1 relations of an Euclidean distance calculation. (b) shows the same ACF with the DTW relations.

B. Segment-123 Algorithm

Since the chance of erroneous user input (also label noise, which again leads to a bias in the classifier training) increases with the length of a trip, we apply SEG-IP in an iteratively and incrementally structured algorithm to facilitate the learning of a robust classifier. The algorithm Segment123 (Algorithm 2) initializes with trips consisting of a single segment to learn mostly clean features and continues iteratively with trips having 2 segments, then 3 and so forth. For each label the clustering is performed separately to learn label-specific patterns. In each iteration step the prediction step follows the clustering to define G , the "good" subsegments (prediction coincides with user-given label) of the clustered ones and those of trips with one more segment. In turn for the next clustering step only the good subsegments are clustered to neglect biased patterns. The clustering method used is partitioning around medoids (PAM [17]), which initializes with a set of medoids, and assigns the data to the closest medoids. By iteratively swapping non-medoids data with medoids the possibly best choice of clusters and medoids is found. This algorithm has two main advantages contrary to traditional k-means clustering. Firstly it is more robust against outliers, and secondly it allows a similarity measure that does not fulfill the triangle inequality, which is required in this case, since DTW does not. The number of clusters is defined with the need of the well known silhouette index [18] which puts into relation a data points distance to its assigned cluster and its distance to the other clusters. Some notation for algorithm Segment123:

- S_t is the set of all segments of trip t , $S(T) = \{S_t | t \in T\}$.
- $S^l(T)$ is the set of segments $\in S(T)$ with label l .
- $\Sigma(t) = \{\sigma \in t\}$; $\Sigma(T)$, $\Sigma^l(T)$ are defined analogously.
- $T_i = \{t \in T : |S_t| = i\}$ is the set of all trips having i -many segments.

The algorithm Segment123 is applied on a set of trips T with segments S and subsegments Σ to learn clusters, used in

Algorithm 2 The algorithm Segment123 trains the S123-NMC based on subsegments

```

1: procedure SEGMENT123( $T, S, \Sigma$ )
2:    $nos \leftarrow 1$  ▷ number of segments
3:    $G \leftarrow \Sigma(T_{nos})$  ▷ initiate G, good subsamples
4:   while  $T_{nos} \neq \emptyset$  do
5:     for  $l$  in  $L$  do ▷ L = set of all labels
6:        $\Sigma^l(G)$ 
7:        $\text{cluster } \Sigma^l(G)$ 
8:        $\text{get cluster Medoids}$ 
9:     end for
10:     $\Sigma^* \leftarrow \Sigma(\bigcup_{j \leq nos+1} T_j)$ 
11:    for  $\sigma \in \Sigma^*$  do
12:       $\hat{l}_\sigma \leftarrow \text{label of closest Medoid}$  ▷ Prediction
13:    end for
14:     $G \leftarrow \{\sigma \in \Sigma^* | l = \hat{l}\}$  ▷ update G
15:     $nos \leftarrow nos + 1$ 
16:  end while
17:  S123-NMC  $\leftarrow$  set of Medoids
18:  return S123-NMC
19: end procedure

```

a NMC, called **S123-NMC**. That classifier in turn is used to predict the labels of new time series.

IV. EXPERIMENTS

A. Evaluation Design

We test the classifier S123-NMC against a model without prior breakpoint detection, employing the user-given segments as they are, that is per label the segments are clustered by the DTW distance of their ACF feature vectors. The cluster medoids are predestined as representatives in a NMC, notated as Segment-NMC (**S-NMC**). For the evaluation phase no user-given breakpoints are assumed. So we adjust the algorithm SEG-IP to a segmentation with no prior knowledge (**SEG-NP**), where the fitness function is applied to a trip as sliding window function. We set the window size to 1 minute which is found to give a reasonable trade off between granularity and the risk of selecting too wide windows where possibly more information is lost by averaging. From the recorded trips 200 were drawn randomly and split into 100 for training the classifiers and 100 for evaluation. This procedure was repeated 200 times and the results were aggregated. For the synthetic case the procedure was repeated 20 times for five different values of δ .

For each of the subsegments achieved from algorithm SEG-NP the label is known. However, unlike to the training case, here it is possible that a subsegment is not fully covered of the real segment, because the user-given information is excluded in the breakpoint detection. To predict a segments label, the predicted labels of subsegments σ having a non-empty intersect with the target segment s are aggregated by a weighted majority voting, where the weight per σ is relative to the length of its intersect with s . This means the bigger the ratio $\frac{|\sigma \cap s|}{|s|}$ the more important σ is for the prediction of the label of s .

B. Data

1) *Synthetic Data*: Each simulated trip consists between 1 to 5 labeled ($l \in \{A, B, C, D\}$) segments ($N = 5000$),

each simulated by an autoregressive (AR) process plus a zero mean random component. Table I gives the corresponding AR-coefficients and the standard deviation (sd_{ar}) of the error term. In real-world data walking segments show a very characteristic ACF, and moreover, each change between transport modes can be assumed to involve walking, so that every second segment can be assumed to be a walking segment. In order to generate synthetic data with similar properties we select appropriate AR coefficients for class A and ensure that every second segment is of class A, beginning randomly with the first or second segment. Further a segmentation bias is simulated by attaching a vector of biased breakpoints to each trip. The breakpoint bias is calculated as a Gaussian random variable with zero mean and a standard deviation sd_b to be varied for different scenarios ($sd_b = |s| \cdot \delta$). The parameter δ varies in a range of 0.1 to 0.5.

Label:	A	B	C	D
AR1	0.9	0.99	0.5	0.2
AR2	-0.9	0	-0.5	0
sd_{ar}	2	1	0.5	0.5

Table I: Coefficients to simulate the synthetic AR processes

2) *Real-World Accelerometer Data*: The data used in this study were collected in a smartphone assisted Prompted Recall survey by 70 volunteers under realistic circumstances and during normal daily activities during four months. The trips were recorded with a logging app installed on the volunteers private Android smartphones, which transmitted the data to a server. No instructions were given on how to carry the phone or which routes and transport modes to choose. The collected data includes GPS positions sampled at 1Hz and the readings of a 3D-accelerometer at 100Hz. Positions and acceleration measurements are synchronized based on timestamps associated with each acceleration and position reading. The participants manually segmented each trip and annotated the transport mode used in each segment by accessing their recorded trips via a web-application, where each trip was displayed on a map. The participants indicated the changes between different modes of transport either by choosing a location on the map or by specifying a point in time. In total 386 hours of trip data were collected with transport mode shares given in Table II. For this study we only used acceleration data of trip segments labeled as walk, car, bus, tram, metro, or train, and we subsumed the latter three into a common category "rails". A typical accelerometer sensor

	WALK	CAR	BUS	TRAM	SUBWAY	TRAIN
Hours	212	81	37	19	21	16
Number	1140	257	209	125	152	58

Table II: Amount of collected real-world trips

built in smart phones (or smart watches or similar wearables) collects 3-dimensional signals (x,y,z). Preprocessing is necessary to assign a measurement of any of the 3 axes to a direction in the real-world, since the orientation of the device is unknown. Since no information about additional sensors was collected (as e.g. magnetometer or gyroscope which would

increase the battery consumption [7]) the 3 dimensional time series is projected to a 2-dimensional (horizontal, vertical) as described in [7]. Estimating the gravity component (the accelerometer sensor permanently measures the gravity force) gives an "anchor" vector which is known to direct to the middle of the earth. Further the admission control is applied as described in [10] to recognize device rotations, which supports the gravity component estimation.

C. Results

1) *Synthetic Data*: In the following we compare the two models (S-NMC the model learned based on the user-given segments, and S123-NMC learned on subsegments with algorithm Segment123) on synthetic data. The confusion matrices in Tables III and IV average the results (so these are not integers) of all random draws and give an overview of the simulated and predicted labels to demonstrate how S123-NMC outperforms S-NMC. Especially the mixup with category A for the S-NMC is due to biased feature learning. S123-NMC compensates this bias mostly, which is consistent with the example in the Figures 1, 2 and 3.

The number of found clusters varies per label and random draw in the range between 2-6. Figure 5 depicts the learned medoids of the two classifiers (S-NMC in the upper row (a) and S123-NMC in the lower row (b)) for a single random draw with $\delta = 0.5$ (for this draw 2-3 clusters were found per label). Here the evolution of the simulated segmentation bias finds its final state in affecting the cluster representatives of the NMCs. It can be seen that for S-NMC the typical pattern of A (resembling a walking pattern) biases the clusters of the other labels, whereas the new segmentation helps to filter out this bias for learning the medoids for S123-NMC. The red dashed lines represent the theoretical ACF corresponding to the values of Table I, which are matched closely by the learned medoids of S123-NMC.

To analyze the effect of the degree of a segmentation bias the parameter δ was varied in a range of 0.1 to 0.5, where for each value 20 different synthetic datasets were simulated. Figure 6 gives the aggregated performance measures for both models in terms of the Fscore¹ [19]. Of course the models perform better the smaller δ but also the differences of the models' Fscores increase with higher δ (5% difference of the Fscores for $\delta = 0.1$ and 17% for $\delta = 0.5$).

Prediction	Simulation					Simulation			
	A	B	C	D		A	B	C	D
A	70.0	11.5	0.7	0.8	A	152.3	4.6	1.9	5.0
B	19.1	46.5	0.0	0.0	B	22.2	53.1	0.0	0.0
C	48.3	0.0	61.8	1.1	C	5.4	0.2	61.1	0.0
D	46.6	0.0	0.5	58.1	D	4.1	0.1	0.0	55.0

Table III: Confusion matrix for model S-NMC

Table IV: Confusion matrix for model S123-NMC

¹The Recall is the average of the share of correct predictions relative to the number of observations per label, and the Precision is the average of the share of correct predictions relative to the number all predictions per label. The overall measure of performance is the FScore, which is the harmonic mean of Recall and Precision.

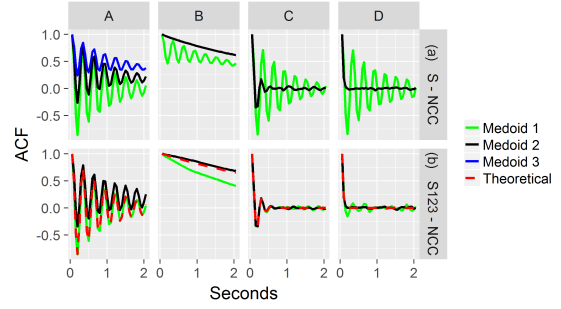


Figure 5: Medoids of the two NMC Models for Synthetic Data for one of the random draws with a δ of 0.5. The red dashed lines are the theoretic ACF of the simulations.

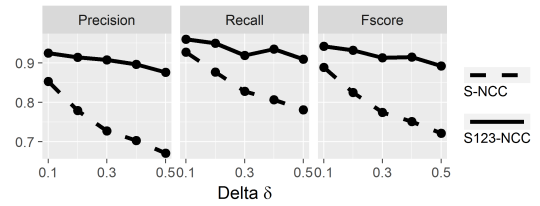


Figure 6: Results of simulated data sets, 20 runs for each δ

2) *Real-World Data*: Tables (V to VII) provide details how the proposed method S123-NMC outperforms S-NMC and show that the Fscore improves contrary to the baseline model. As in the synthetic case the learned features from the two classifiers differ significantly. Figure 7 depicts the medoids of the two classifiers (in the upper row (a) S-NMC and in the lower row (b) S123-NMC). One can recognize that in (a) the typical walking pattern of high frequency and magnitude biases the ACF of car, bus and rail (the 1st medoids per label, in green). However in (b) this bias is vanished. Another biased feature vector is the third medoid of walk (blue line in (a)) that is probably learned falsely from a mixup of walk and rail. This bias is also corrected in (b), that is the algorithm Segment123 did not learn this feature vector to be a representative for walk.

Model	Precision	Recall	Fscore
S123-NMC	0.48	0.50	0.49
S-NMC	0.46	0.47	0.46

Table V: Comparison of results for real-world data

Prediction	User-Input	User-Input				Prediction	User-Input	User-Input			
		W	B	C	R			W	B	C	R
WALK	WALK	42.0	0.6	0.4	3.1	WALK	WALK	44.3	0.4	0.4	3.6
	CAR	4.3	18.6	3.4	2.5		CAR	4.2	20.3	3.5	2.4
	BUS	8.6	13.5	3.7	2.4		BUS	4.1	10.8	3.3	1.4
	RAIL	3.4	1.0	0.6	1.6		RAIL	5.7	2.2	0.9	2.1

Table VI: Confusion matrix for model S-NMC

Table VII: Confusion matrix for model S123-NMC

Precision and Recall for the real-world dataset are fairly low. However, these figure express performance in relation to a ground dataset, and we do not have knowledge about the actual ground truth data. Instead, the reference data are biased due to

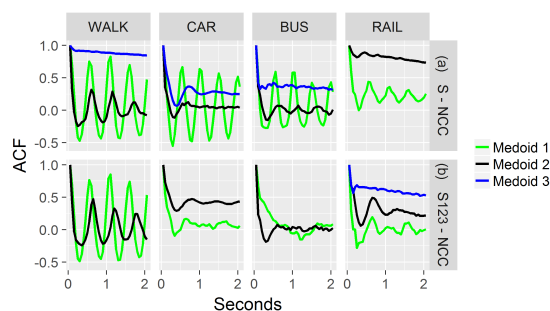


Figure 7: Medoids of the two NMC Models for real-world data for one of the random draws.

erroneous user input. Achieving a high level of performance is hardly possible in case of extensively biased segments, where the most important representative subsegments have different ground truth labels than the users stated. Imagine a trip as given in Figure 3 with a subway trip so short such that the subsegment σ_4 is shorter than the sum of the enclosing biasing subsegments. The label of the subway segment s_2 would then be impossible to predict correct due to the extensive bias. Tables VI and VII must be therefore interpreted with care. In addition, we want to stress that in relation to other approaches in the literature our approach does not use location data and relies only on accelerometer readings.

Nevertheless we proved our method to learn cluster medoids – based on the homogeneous subsegments – that are revised from a bias introduced by false segmentation (Figure 7). Therefore employing the proposed method in data collection applications will improve the classification performance as proven for the synthetic case.

V. CONCLUSION

This paper showed that training a classification model for transport mode recognition with noisy ground truth data obtained from raw user input can adversely impact the classifier’s performance. In particular we showed that aside from label noise, another reason for this phenomenon is segmentation bias: inaccurate segmentation of the trip into segments. We proposed a method to correct user-supplied breakpoints by identifying changes in the signal’s variance to increase the homogeneity within each trip segment. We showed experimentally with both synthetic and real-world data, that applying the proposed method reduces segmentation bias, resulting in better separability between the modeled classes and improved classification performance.

The proposed method for segmentation correction can be incorporated in machine learning models to improve the performance of activity recognition models and transport mode classifiers by providing more homogeneous data for both model calibration and prediction.

For future work we plan to collect real ground truth data in a more elaborated setup with researchers having a genuine interest in data quality who will indicate breakpoints between travel modes during travel. This will resolve the issue of

erroneous ground truth data collected by survey respondents and provide more realistic performance measures. Data fusion methods will be developed to learn both from the real world dataset used in this paper and the newly collected dataset. Further we will focus on entangling the breakpoint detection and clustering steps to investigate possibly iterative improvements in homogeneity of the single segments and the clusters.

REFERENCES

- [1] P. Nitsche, P. Widhalm, S. Breuss, N. Brändle, and P. Maurer, “Supporting large-scale travel surveys with smartphones - A practical approach,” *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 212–221, 2014.
- [2] A. Ellison, R. Ellison, D. Rance, C. Standen, C. Rissel, and M. Crane, “A Web-Based Diary and Companion Smartphone app for Travel/Activity Surveys,” *Transportation Research Procedia*, vol. 11, pp. 297–310, 2015.
- [3] P. R. Stopher, L. Shen, W. Liu, and A. Ahmed, “The Challenge of Obtaining Ground Truth for GPS Processing,” *Transportation Research Procedia*, vol. 11, pp. 206–217, 2015.
- [4] C. Carrion, F. Pereira, R. Ball, F. Zhao, Y. Kim, K. Nawarathne, N. Zheng, C. Zegras, and M. Ben-Akiva, “Evaluating fms: a preliminary comparison with a traditional travel survey,” in *Transportation Research Board 93rd Annual Meeting*, no. 14-5541, 2014.
- [5] B. Frénay and M. Verleysen, “Classification in the presence of label noise: a survey,” *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845–869, 2014.
- [6] X. Zhu and X. Wu, “Class noise vs. attribute noise: A quantitative study,” *Artificial intelligence review*, vol. 22, no. 3, pp. 177–210, 2004.
- [7] S. Hemminki, P. Nurmi, and S. Tarkoma, “Accelerometer-based transportation mode detection on smartphones,” *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems - SenSys '13*, pp. 1–14, 2013.
- [8] S. Kaghyan and H. Sarukhanyan, “Activity recognition using K-nearest neighbor algorithm on smartphone with Tri-axial accelerometer,” *International Journal of Informatics Models and Analysis*, vol. 1, pp. 146–156, 2012.
- [9] P. Siirtola, P. Laurinen, E. Haapalainen, J. Röning, and H. Kinnunen, “Clustering-based activity classification with a wrist-worn accelerometer using basic features,” *2009 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2009 - Proceedings*, pp. 95–100, 2009.
- [10] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, *Proceedings of the 8th Conference on Embedded Networked Sensor Systems*, pp. 71–84.
- [11] Y. E. Ustev, O. Durmaz Incel, and C. Ersoy, “User, device and orientation independent human activity recognition on mobile phones: Challenges and a proposal,” pp. 1427–1436, 2013.
- [12] P. Widhalm, P. Nitsche, and N. Brändle, “Transport mode detection with realistic smartphone sensor data,” in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 573–576.
- [13] M. Mahdavian and T. Choudhury, “Fast and scalable training of semi-supervised crfs with application to activity recognition,” pp. 977–984, 2008.
- [14] E. Keogh, S. Chu, D. Hart, and M. Pazzani, “An online algorithm for segmenting time series,” in *Proceedings 2001 IEEE International Conference on Data Mining*, 2001, pp. 289–296.
- [15] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H. T. T. Toivonen, “Time series segmentation for context recognition in mobile devices,” in *Proceedings 2001 IEEE International Conference on Data Mining*, 2001, pp. 203–210.
- [16] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, Feb 1978.
- [17] L. Kaufman and P. Rousseeuw, “Clustering by means of medoids,” *Statistical Data Analysis Based on the L1 Norm and Related Methods*, edited by Y. Dodge, North-Holland.
- [18] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987.
- [19] D. M. W. Powers, “Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation,” *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

Paper B

Automatic Detection of Warped Patterns in Time Series: The Caterpillar Algorithm

License Information

©2018 IEEE. Reprinted, with permission, from Leodolter et.al., Automatic Detection of Warped Patterns in Time Series: The Caterpillar Algorithm, 2018 IEEE International Conference on Big Knowledge (ICBK), 2018.

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Vienna's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

Automatic Detection of Warped Patterns in Time Series: The Caterpillar Algorithm

Maximilian Leodolter
 Center for Mobility Systems
 Austrian Institute of Technology
 Vienna, Austria
 maximilian.leodolter@ait.ac.at

Norbert Brändle
 Center for Mobility Systems
 Austrian Institute of Technology
 Vienna, Austria
 norbert.braendle@ait.ac.at

Claudia Plant
 Faculty of Computer Science
 ds:UniVie University of Vienna
 Vienna, Austria
 claudia.plant@univie.ac.at

Abstract—Detection of similar representations of a given query time series within longer time series is an important task in many applications such as finance, activity research, text mining and many more. Identifying time warped instances of different lengths but similar shape within longer time series is still a difficult problem. We propose the novel Caterpillar algorithm which fuses the advantages of Dynamic Time Warping (DTW) and the Minimum Description Length (MDL) principle to move a sliding window in a crawling-like way into the future and past of a time series. To demonstrate the wide field of application and validity, we compare our method against state-of-the-art methods on accelerometer time series and synthetic random walks. Our experiments demonstrate that Caterpillar outperforms the comparison methods in detecting accelerometer signals of metro stops.

Index Terms—Minimum Description Length, Dynamic Time Warping, Accelerometer Data

I. INTRODUCTION

Time series data are ubiquitous across many fields such as finance, medicine and activity research. Figure 1a shows an example of a time series, illustrating the accelerometer signal recorded by a person traveling on a metro while carrying a smartphone. Intuitively, the red colored segments resemble the given query pattern in Figure 1b, which represents a typical braking-stopping-accelerating sequence at a metro stop. Further it is obvious that in between the stations the signal has a high variation and follows no distinct pattern. However, it is difficult to detect these vertically shifted and time warped instances of the query pattern.

We propose a novel technique solving the above problem and with (1) a novel modeling approach combining the Dynamic Time Warping (DTW) and Minimum Description Length (MDL) principle and (2) a novel algorithm – the Caterpillar – efficiently detecting time warped fits of model time series within longer time series. The proposed model encodes a time series conditionally on a time warped representation. Using simple matrix transformations the algorithm moves a sliding window, similar to a caterpillar extending and contracting to crawl. Figure 1a illustrates the detected metro stops of different time extension (red numbers) and vertical scale.

When comparing two time series, the length of the query pattern and the time extension of typical shapes often vary,

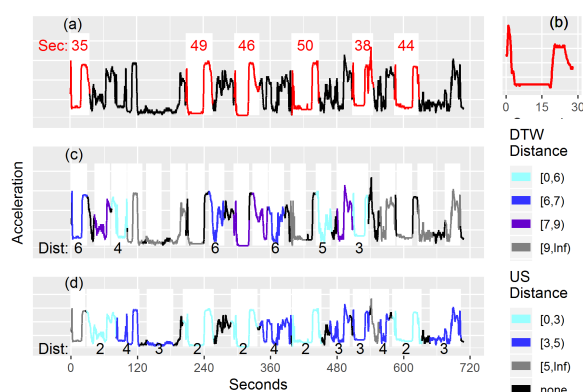


Figure 1: Horizontal accelerometer signal component of a person traveling on a metro while carrying a smartphone. In (a) the Caterpillar algorithm detected the red colored time warped (duration in seconds is plotted) and vertically shifted instances of the query pattern in (b). In (c) the fits found by the baseline method UCR Suite DTW and the respective distances, and the equivalent for UCR Suite US in (d).

such as the duration of the stop or braking in the metro smartphone example above. In many cases it is therefore crucial to consider varying length and time extension when selecting an appropriate distance measure for time series analysis tasks such as pattern detection, classification and clustering. Common distance measures between time series include Euclidean distance, Fréchet distance, SAX [13], Uniform Scaling (US) [15], edit distances, Dynamic Time Warping (DTW) [17], and many more. DTW allows a non-linear alignment of two time series and can thus match them in case of similar shape, yet different time extensions.

Time series data analysis often boils down to the binary question of whether a given time series sample fits to a query time series or not. Traditionally, any distance measure requires setting a threshold parameter. The threshold parameter value is crucial, yet difficult to identify, and in general heavily affects the analysis results. UCR Suite [15] (applying DTW or US) is the state-of-the-art method for scanning longer a time series to detect a query pattern. Given the set of distances to the query pattern of the fits detected by UCR Suite, it is hard to identify

a suitable threshold. Figure 1c and Fig. 1d show the result of matching the query pattern of Figure 1b using UCR Suite with DTW and US distances, respectively. The first segment starting at second 0 with a DTW distance of 6 obviously resembles the query pattern, and the one starting at second 430 with a lower DTW distance clearly not. The same applies to the detected segments and their US distances in Fig 1d starting at second 0 and 130 (more details in Sec. V-C).

The MDL principle is a promising tool which does not require identifying such sensitive distance parameter values, and can be considered as a formalization of Occam’s Razor. The main idea is that the best model for a given data set will be the model achieving the highest data compression rate. MDL has been applied in many fields so far (e.g. bioinformatics [20], text mining [5], etc.) for model selection and also for time series analysis [2], [21], [7], [16], [22]. Yet it is still an ongoing process to investigate the potential of MDL for time series analysis. To the best of our knowledge there exists no similar approach combining entropy based MDL and DTW, thus avoiding parameter value setting for distances.

Our main contributions are:

- A novel encoding scheme fusing the MDL principle and DTW to match time warped time series.
- The novel Caterpillar algorithm matching time series in the future and the past to find the best alignment.
- Extensive experiments demonstrate broad applicability on accelerometer time series and synthetic data.

We denote a univariate time series as $\mathbf{x} := \{x_i\}_{i=1\dots m}$ and repeat the definition of the well known Euclidean distance for two time series \mathbf{x} and \mathbf{h} of same length as: $\sqrt{\sum(x_i - h_i)^2}$.

This paper is organized as follows: The next section discusses different stepwise calculations of DTW. Section III proposes the MDL encoding scheme for time series of different lengths that is applied in the Caterpillar algorithm presented in Sec. IV. Section V demonstrates the broad applicability of our method by extensive experiments before we discuss our method in the context of related work in Sec. VI.

II. FAST STEPWISE DTW CALCULATION

The Euclidean distance is not well suited to recognize distorted patterns along the time axis in time series, since it requires a 1-1 alignment between an entry of \mathbf{x} and a single entry of \mathbf{h} . The work in [17] introduced the Dynamic Time Warping (DTW) distance to find non-linear alignments between time series of possibly different lengths. To calculate the DTW we define the matrix of differences Δ , ($\Delta_{ij} = x_i - h_j$) and the cost matrix \mathbf{C} ($C_{ij} = |\Delta_{ij}|$). The cumulative global cost matrix \mathbf{G} is calculated as follows:

$$\mathbf{G}_{i,j} = \begin{cases} \sum_{k \leq i} \mathbf{C}_{k,1} & j = 1 \\ \sum_{l \leq j} \mathbf{C}_{1,l} & i = 1 \\ \mathbf{C}_{i,j} + \min(\mathbf{G}_{i-1,j}, \mathbf{G}_{i,j-1}, \mathbf{G}_{i-1,j-1}) & i, j > 1 \end{cases} \quad (1)$$

with \mathbf{G}_{nm} being the DTW distance measure. The matrix \mathbf{G} is calculated simultaneously with the direction matrix \mathbf{D} ,

$\mathbf{D}_{ij} \in \{1, 2, 3\}$. Each element of \mathbf{D} represents which step is cheapest to take next, either diagonal (1), horizontal (2) or vertical (3). The warping path ω is an excerpt of \mathbf{D} and is the vector of steps (diagonal, vertical, horizontal) taken to find the cheapest path from \mathbf{G}_{nm} back to \mathbf{G}_{11} to align the two time series \mathbf{x} and \mathbf{h} with the lowest cumulative costs. The difference path δ is an excerpt of Δ , and constitutes the vector of differences of the elements of \mathbf{x} and \mathbf{h} that are aligned to each other as described by ω . Finally, to represent \mathbf{x} by a given \mathbf{h} , we require information about the deviations (δ) and the alignment (ω) of the time series. Figures 2a-c illustrate the defined matrices and vectors for a simple example.

A. In-/Decremental DTW

Given the result of $DTW(\mathbf{x}, \mathbf{h})$ for \mathbf{h} and $\mathbf{x} = \{x_i\}_{i=1\dots m}$, suppose new observations are appended at the end of \mathbf{x} denoted as $\mathbf{x}^+ = \{x_i\}_{i=1\dots m+k}$. To calculate $DTW(\mathbf{x}^+, \mathbf{h})$ we can append the cumulative costs of new observations according to (1) to the already computed \mathbf{G} and receive \mathbf{G}^+ . Figure 2d shows the incremental step to include a further observation of \mathbf{x} , 7 to the alignment to \mathbf{h} .

The opposite of the incremental step is to omit potentially irrelevant observations at the end of \mathbf{x} and recalculate the DTW distance given the results of the complete case. The matrices Δ , \mathbf{C} and \mathbf{G} can be updated directly by omitting the respective columns at the end. Figure 2e shows the decremental step to exclude the final observation of \mathbf{x} , 7 from the alignment to \mathbf{h} .

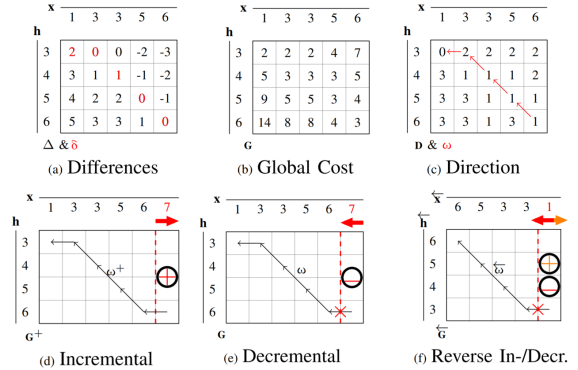


Figure 2: DTW computation: Based on Δ (a) the matrix \mathbf{G} (b) and \mathbf{D} (c) are computed in parallel. Next the paths ω (c) and δ (a) are extracted from the respective matrices by tracking back the given steps in \mathbf{D} . Also the incremental inclusion (d) of a new observation of \mathbf{x} , 7 (e) the decremental exclusion of the last element, 7 and (f) the reverse-decremental exclusion of the first element, 1.

B. Reverse In-/Decremental DTW

In-/decremental steps allow to react to alterations of the end of \mathbf{x} . Can we do the same for alterations at the initial observations of \mathbf{x} ? The answer is no, because the basic structure of the DTW algorithm and the dynamic calculation

of \mathbf{G} does not allow to alter, add or omit observations at the beginning (initial/past values of \mathbf{x}) without changing the entire global cost matrix. However [1] proved the DTW distance calculation to be reversible: $DTW(\mathbf{x}, \mathbf{h}) = DTW(\overleftarrow{\mathbf{x}}, \overleftarrow{\mathbf{h}})$, where $\overleftarrow{\mathbf{x}}$ is \mathbf{x} in reverse order: $\overleftarrow{\mathbf{x}} := \mathbf{x}(m : 1)$. We apply this principle and proceed analogously to the incremental (decremental) step to finally add (omit) initial values of \mathbf{x} . Figure 2f depicts the reverse time series after omitting 7. To exclude the first element of \mathbf{x} , 1 we calculate $DTW(\overleftarrow{\mathbf{x}}, \overleftarrow{\mathbf{h}})$. The main advantage of the reverse-in/decremental calculation becomes evident when comparing results of adding or omitting more than one initial element of \mathbf{x} . Since for each step after the initial step, the results from the previous calculation can be recycled analogously to the in-/decremental DTW calculation, again computation time can be saved (details in section IV-E).

C. Vector based Dynamic Time Warping Implementation

If one is interested in only the DTW distance measure and not in the alignment, one can save computation time by not allocating the matrices \mathbf{D} , \mathbf{C} and \mathbf{G} for the DTW calculation. Since the calculation of the j -th column of \mathbf{G} , $\mathbf{G}_{:,j}$ only depends on $\mathbf{G}_{:,j-1}$, instead of allocating a new vector, the vector storing information of $\mathbf{G}_{:,j-2}$ is overwritten with $\mathbf{G}_{:,j}$. Incremental calculations for new observations are possible as well, as long as the previously calculated vector $\mathbf{G}_{:,m}$ is saved. For decremental calculations we allocate an additional vector that stores the last entries of each column, $\mathbf{G}_{n,\cdot}$.

We stress this calculation principle (also implemented in the R packages IncDTW [11] and rucrdtw [15] that we also apply) since we apply it for the fusion of DTW and MDL in Sec. IV-D.

III. DTW-BASED DESCRIPTION LENGTH

To apply the MDL principle we first normalize and discretize a time series \mathbf{x}^0 and define the discrete time series \mathbf{x} , $\forall j \in 1 \dots m$ (also done in e.g. [16], [2], [7], [21], [22]) as:

$$x_j := \text{round}\left(\frac{x_j^0 - \min}{\max - \min} \cdot 2^b - 1\right) + 1. \quad (2)$$

\min and \max are the minimum and maximum values of \mathbf{x}^0 . For discretization of a set of time series, \min and \max are replaced by global equivalents. The value of b is set to 6 which hardly affects the accuracy of common time series analysis tasks (such as clustering or classification) as discussed in [16].

A. Conditional Description Length

We define the number of required bits to encode a time series of length m as the description length:

$$DL(\mathbf{x}) = -\sum_{i=1}^m \log_2(P(x_i)), \quad (3)$$

where P is the probability function, discussed in detail in Sec. III-B. We apply the two part MDL principle to describe the conditional description length and estimate the description length by the entropy of the time series. According to Kraft's inequality, the length of any lossless code is lower bounded

by the entropy. In case the hypothesis \mathbf{h} and \mathbf{x} are of the same length, the code length of \mathbf{x} conditional on the hypothesis \mathbf{h} is defined similar to their Euclidean distance (in the following called **E-model**):

$$DL(\mathbf{x}, \mathbf{h}) := DL(\mathbf{h}) + DL(\mathbf{x} | \mathbf{h}) = DL(\mathbf{h}) + DL(\mathbf{x} - \mathbf{h}). \quad (4)$$

We call $DL(\mathbf{x}, \mathbf{h})$ from (4) the conditional description length (CDL) of \mathbf{x} given \mathbf{h} according to the **E-model**. For time series of different lengths and/or different extensions in time of typical patterns, we propose to incorporate the DTW algorithm into the MDL principle. As discussed in Sec. II, the DTW algorithm not only calculates the DTW distance measure, but also the warping path ω and the path of differences δ . We use these components to incorporate the main advantage of DTW allowing non-linear alignments of time series of different lengths and define the DL for \mathbf{x} conditional on \mathbf{h} (and call it **DTW-model**):

$$\begin{aligned} DL(\mathbf{x}, \mathbf{h}) &:= DL(\mathbf{h}) + DL(\mathbf{x} | \mathbf{h}) \\ DL(\mathbf{x} | \mathbf{h}) &:= DL(\delta) + DL(\omega) = DL(\delta) + |\omega| \lceil \log_2 3 \rceil. \end{aligned} \quad (5)$$

Comparing (4) and (5), $DL(\mathbf{x} - \mathbf{h})$ is the equivalent of $DL(\delta)$. The additional warping costs $DL(\omega)$ trade off the possibly lower DL of differences due to non-linear alignments. As every direction in the path is about equally frequent, we use 2 bits to encode each element.

Given a set of time series \mathbf{X} and a hypothesis \mathbf{h} , a time series \mathbf{x} is a candidate-match if $DL(\mathbf{x}) > DL(\mathbf{x} | \mathbf{h})$. If the sum of the saved costs for all candidate-matches is larger than $DL(\mathbf{h})$, then the hypothesis \mathbf{h} is accepted and the candidate-matches are considered as true matches. Otherwise it is cheaper to describe candidate-matches unconditionally.

In general, two time series to be compared are not of the same length. Furthermore, depending on the application, it might be preferable not to match the entire time series \mathbf{x} to a given hypothesis \mathbf{h} , but to consider only a segment of \mathbf{x} as a valid match to \mathbf{h} . Since the logarithm is additive we can easily split a time series $\mathbf{x}(1 : m)$ into disjunct segments and describe different segments independently from each other, either unconditionally or conditionally to \mathbf{h} . If multiple fits are found, the saved DL is the difference between $DL(\mathbf{x})$ and the sum of encoding costs for the conditional and unconditional segments. For lossless compression, we also need to encode the information of the starting indices, ι of the detected candidate fits. The index costs are defined as:

$$DL(\text{indices}) := \sum_{i \geq 1} \lceil \log_2(m - \xi_{i-1}) \rceil,$$

where $\xi_0 = 0$. The index costs for the second fit are smaller than those of the previous, since we make use of $\iota_2 > \xi_1$. If $DL(\text{indices})$ is bigger than the saved DL, the candidate fits are discarded.

B. Probability Density Function

The applied probability density function (PDF) in (3) is supposed to be as general as possible and applicable in many

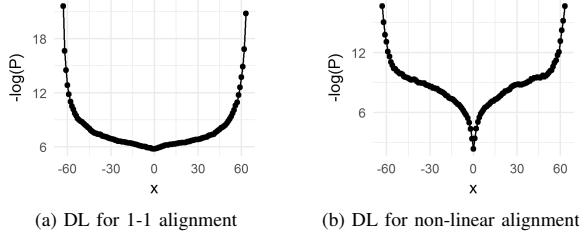


Figure 3: DL of differences: Logarithms of the estimated Probability functions.

different domains and at the same time provide valid approximations of the intrinsic PDFs. In contrast to the approaches of [16], [22], we apply global PDFs for our models instead of estimating P directly from each time series segment, since we intensively stress the additivity of the CDL. In the following we describe how we selected the PDF and Sec. V will demonstrate the broad field of application for our choice. We choose to apply a uniform PDF for the unconditional DL, so that $DL(\mathbf{x}) = |\mathbf{x}| \cdot b = |\mathbf{x}| \cdot 6$. This way the DL depends only on the length of a time series and not on its values, which implicates that the decision if \mathbf{x} is described best by \mathbf{h} depends only on the differences and not on the values of \mathbf{x} itself.

Further we apply different PDFs for linear 1-1 alignments of time series of same lengths and non-linear alignments of time series of possibly unequal lengths. For the E-model (4) we simulate 1000 couples of random walks (\mathbf{x}, \mathbf{h}) , each of length $= 10^6$ independent from each other, discretize them and count the occurrences of the difference vectors $(\mathbf{x} - \mathbf{h})$. For the DTW-model (5) we generate 1000 couples of random walks (\mathbf{x}, \mathbf{h}) independent from each other. The length m of \mathbf{x} is chosen randomly between 100 and 1000, and the length n of \mathbf{h} is drawn randomly out of the interval $[0.5 \cdot m, \dots 2 \cdot m]$. This dependency between the lengths avoids unreasonable combinations. We then calculate DTW and the path of differences δ and count the frequencies per possible value. Figure 3 shows the negative logarithms (DL for a single element of difference) of the two populations. The non-linear alignment clearly causes a higher density – equivalent to lower encoding costs – around 0 than the linear 1-1 alignment.

IV. CATERPILLAR

Given two time series \mathbf{x} and \mathbf{h} of length m and n , with $m > n$, we want to scan the longer time series \mathbf{x} to recognize possibly warped instances of the hypothesis-pattern \mathbf{h} as part of it. Before proposing our method – the Caterpillar algorithm – we mention the baseline approach (inspired by [16] and adapted here by applying a global P instead of segmentwise since we make use of the additivity of the DL) applying the E-model from (4) and denote it as PREMDL (Pattern recognition by Euclidean distance and the MDL). For each index $i \in 1 \dots m - n$, we calculate $DL(\mathbf{x}(i : i + n), \mathbf{h})$ and evaluate if the following two conditions $\forall j \in \{i - n, \dots, i + n\}$ hold to

find a match.

$$DL(\mathbf{x}(i : i + n), \mathbf{h}) < DL(\mathbf{x}(i : i + n))$$

$$DL(\mathbf{x}(i : i + n), \mathbf{h}) < DL(\mathbf{x}(j : j + n), \mathbf{h}).$$

A. The Caterpillar Algorithm

We propose the following Caterpillar algorithm to scan a longer time series \mathbf{x} to detect a warped instance of \mathbf{h} . Given a couple of starting indices ι (lower) and ξ (upper) the Caterpillar algorithm performs one of the following movement steps:

- Forward: Increase ξ : $\xi^+ > \xi$; evaluate $DL(\mathbf{x}(\iota : \xi^+), \mathbf{h})$
- Back up: Decrease ξ : $\xi^- \leq \xi$; evaluate $DL(\mathbf{x}(\iota : \xi^-), \mathbf{h})$
- Catch up: Increase ι : $\iota^+ \geq \iota$; evaluate $DL(\mathbf{x}(\iota^+ : \xi), \mathbf{h})$
- Backward: Decr. ι : $\iota^- < \iota$; evaluate $DL(\mathbf{x}(\iota^- : \xi), \mathbf{h})$.

The CDL is calculated according to the DTW-model (5). These 4 steps enable the algorithm to move the scanning window $(\iota : \xi)$ along \mathbf{h} similarly to a caterpillar that is extending to move forward or backward, and can also contract its front to back up, or its back to catch up.

Each of the movement steps requires to compute the DTW. To save computation time we calculate the DTW according to the computation methods proposed in Sec. II: incremental for forward, decremental for back up, reverse-decremental for catch up, and reverse-incremental for backward. Figures 2d-f give examples for the respective calculations. Finally a match is found if:

$$DL(\mathbf{x}(\iota : \xi) | \mathbf{h}) < DL(\mathbf{x}(\iota : \xi)). \quad (6)$$

Algorithm 1 gives details on how these movements help the Caterpillar algorithm to detect warped patterns similar to \mathbf{h} in \mathbf{x} . Figures 4a-d depict the iterative movements of the Caterpillar for a simple example. The Caterpillar also helps to decide whether two time series should be matched completely (Fig. 4b), or rather partially (Fig. 4d).

B. Find initial lower index ι

Because of the additivity of the DL definition we can initiate the Caterpillar algorithm in parallel on different initial indices ι . To find initials for the Caterpillar algorithm we apply the PREMDL algorithm. The first index of a found match is selected to be an initial lower index ι . Depending on the application, the proposed algorithms can also detect matches in \mathbf{x} of the same shape as \mathbf{h} , however vertically shifted for a fixed shift α (compare 6):

$$DL(\mathbf{x}(\iota : \xi), \mathbf{h} + \alpha) + DL(\alpha) < DL(\mathbf{x}(\iota : \xi)). \quad (7)$$

For a given index ι we define α implicitly when the PREMDL algorithm calculates $DL(d)$ where $d = \mathbf{x}(\iota : \iota + n) - \mathbf{h}$. We define $\alpha := \text{mean}(d)$.

C. Lower Bound for CDL: Find initial upper index ξ

In case $|\mathbf{h}| = n > m = |\mathbf{x}|$ we estimate a lower bound for the CDL for a complete match (\mathbf{x} is matched from begin to end to \mathbf{h}):

$$DL(\mathbf{x}, \mathbf{h}) = DL(\omega) + DL(\delta) + DL(\mathbf{h}) \geq DL(\omega) + DL(\delta)$$

$$\geq |\omega| \cdot \lceil \log_2 3 \rceil - |\delta| \cdot \log_2 p(0) \geq n(2 - \log_2 p(0))$$

The first inequality follows from omitting the costs for the hypothesis, which is certainly positive. Also, in case of many matches, the costs are not considered for the calculation of a single match. The second inequality holds, since we suppose the PDF for the warping path to be uniform and the cheapest path for the differences is the one of zeros. The third inequality holds, since the shortest warping path is the one with equal length as the longer time series, \mathbf{h} . So we can combine:

$$DL(\mathbf{x}) = 6m \stackrel{?}{\leq} n(2 - \log_2 p(0)) \leq DL(\mathbf{x} | \mathbf{h}) \quad (8)$$

As long as inequality (8) holds no CDL for the combination of \mathbf{x} and \mathbf{h} needs to be calculated. In case of the proposed distributions of Fig. 3b we abandon calculations as long as $m \leq n \frac{1}{8} (2 - \log_2 p(0)) \approx n \cdot 0.73$. Finally we use this lower bound to determine the initial extension of the Caterpillar, the initial index ξ (which is also the minimum extension of the Caterpillar) for a given ι , respectively:

$$\mu := 0.73 \cdot n + \iota. \quad (9)$$

If \mathbf{x} is much longer than \mathbf{h} , only a segment of the time series (\mathbf{y}) is matched to the hypothesis and the remaining segment (\mathbf{z}) that is described unconditionally. The just derived estimation of the lower bound holds also for this case, since:

$$DL(\mathbf{x}) = DL(\mathbf{y}_i) + DL(\mathbf{z}_i) \stackrel{?}{\leq} DL(\mathbf{y}_i | \mathbf{h}) + DL(\mathbf{z}_i) \quad (10)$$

$$\iff DL(\mathbf{y}_i) \stackrel{?}{\leq} DL(\mathbf{y}_i | \mathbf{h})$$

Due to the additivity of the DL, it does not matter how long the time series \mathbf{x} is. The lower bound estimation only depends on the segment to be matched, \mathbf{y} .

D. Accelerate The Caterpillar

According to the DTW-model (5), the DL is determined by these two steps:

- 1 calculate the DTW, and
- 2 determine the DL on top of the results of step 1.

We fuse these two components to enhance the DTW-model and the Caterpillar algorithm. That is, the DL is used to define the cost matrix \mathbf{C} for the DTW algorithm. In detail:

$$\mathbf{C}_{i,j} := DL(x_i - h_j) = -\log_2 P(x_i - h_j) \quad (11)$$

Then \mathbf{C} is used as input for the DTW algorithm to calculate \mathbf{G} as usual and the calculated DTW distance equals $DL(\delta)$. To take care of the additional costs for the warping path, we could trace back \mathbf{G} to get the warping path ω . However, since we are only interested in the costs of warping, but not ω itself, we skip the interim stage of calculating ω and add the warping costs in the calculation of the global cost matrix. This way we save computation time by not backtracking the global cost matrix to find the warping path. We define the global cost matrix as follows (compare 1):

$$\mathbf{G}_{i,j} = \begin{cases} \sum_{k \leq i} \mathbf{C}_{k,1} + i \cdot 2 & j = 1 \\ \sum_{l \leq j} \mathbf{C}_{1,l} + j \cdot 2 & i = 1 \\ \mathbf{C}_{i,j} + 2 + \min(\mathbf{G}_{i-1,j}, \mathbf{G}_{i,j-1}, \mathbf{G}_{i-1,j-1}) & i, j > 1 \end{cases} \quad (12)$$

We call the $DL(\mathbf{x}, \mathbf{h})$ calculated by the DTW distance between \mathbf{x} and \mathbf{h} based on (11) and (12) the fusion-DTW-model (**fdTW-model**).

Since we are not interested in \mathbf{G} itself, but only the last entry in the last row, we apply the principle of the vector based DTW implementation II-C and additionally save the last column for incremental steps and the last row for decremental steps.

Algorithm 1 sketches the Caterpillar algorithm. As long as the end (ξ^{max}) is not reached and one of the four movements achieves to decrease the CDL, the Caterpillar moves. Each forward while loop increases ξ until the CDL increases. Then the back up step potentially decreases ξ in case the last forward step was too far and the current CDL value is below the $DL(x)$. Next, if no catch up has been performed yet, the Caterpillar has the option to move backward until the start is reached (ι^{min}) or the CDL increases. Finally, the Caterpillar catches up. Without any prior knowledge, we set the initial indices $\iota^{min} = \iota = 1$, $\xi^{max} = |\mathbf{x}|$ (or restrict them by neighboring initials, respectively) and $\xi = \mu$ (see 9).

Algorithm 1 Caterpillar

```

1: PREMDL defines the tuples  $(\iota, \xi, \iota^{min}, \xi^{max})$ 
2: for each tuple call Caterpillar:
3: procedure CATERPILLAR( $\mathbf{x}, \mathbf{h}, \iota, \xi, \iota^{min}, \xi^{max}$ )
4:   CatchUpSuccess  $\leftarrow$  FALSE
5:    $CDL \leftarrow DL(x)$ 
6:   while  $\xi < \xi^{max} \wedge CDL \searrow$  do  $\triangleright$  as long as CDL decreases
7:     repeat
8:        $(CDL, \iota, \xi) \leftarrow$  Forward( $\iota, \xi$ )  $\triangleright$  increase  $\xi$  to decrease CDL
9:     until  $\xi = \xi^{max} \vee CDL \nearrow$ 
10:    if  $CDL < DL(x)$  then
11:       $(CDL, \iota, \xi) \leftarrow$  BackUp( $\iota, \xi$ )  $\triangleright$  decrease  $\xi$  to decrease CDL
12:    end if
13:    if !CatchUpSuccess then
14:      repeat
15:         $(CDL, \iota, \xi) \leftarrow$  Backward( $\iota, \xi$ )  $\triangleright$  decrease  $\iota$  to decrease
        CDL
16:      until  $\iota = \iota^{min} \vee CDL \nearrow$ 
17:    end if
18:     $(CDL, \iota, \xi) \leftarrow$  CatchUp( $\iota, \xi$ )  $\triangleright$  increase  $\iota$  to decrease CDL
19:    if CatchUp successful then
20:      CatchUpSuccess  $\leftarrow$  TRUE
21:    end if
22:  end while
23:  return  $(\iota, \xi)$   $\triangleright$  return the indices of smallest CDL
24: end procedure

```

Fusing DTW and MDL (12) saves computation time by omitting the backtracking step to find the warping path, and enables fast versions of the two Caterpillar movements: back up step and catch up step. To find the optimal index, we compare all possible values since these are stored in the already computed vectors storing information of the last row and last column of the global cost matrix. In detail:

a) *Back Up*: \mathbf{G} is already computed¹ for $\mathbf{x}(\iota : \xi)$ and $\mathbf{h}(1 : n) = \mathbf{h}$, and w.l.o.g say $\iota = 1$. Suppose the last forward movement of the Caterpillar increases the total costs. To evaluate possible back up step widths $j \in \{0, \dots, \xi - \mu\}$, we compare $\mathbf{G}(n, \iota - \xi + 1 - j) + 6j = \mathbf{G}(n, \xi - j) + 6j$.

¹Actually we never compute the whole matrix \mathbf{G} , but only the last column and row, but for simplicity we stick to this notation.

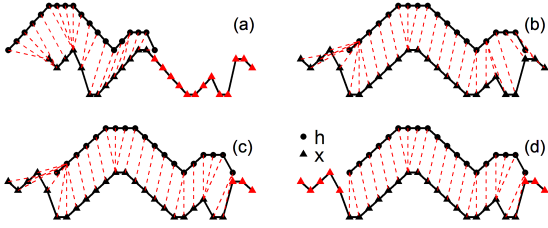


Figure 4: The Caterpillar: After the initial alignment (a) the forward movement concludes in the complete matching (b). Backing up helps to exclude the final 3 observations of \mathbf{x} (c), and finally the catch up in (d) shows the best alignment.

In Fig. 4a the Caterpillar algorithm first aligns the time series \mathbf{x} and \mathbf{h} for the indices $\iota = 1$ and $\xi = \iota + 0.73 \cdot n$. Next the Caterpillar moves forward until $\xi = m$, so the end is reached (Fig. 4b), before it decides to back up three steps ($j = 3$) and represent the last three elements of \mathbf{x} unconditionally (Fig. 4c).

b) *Backward/ Catch Up:* To evaluate possible catch up and backward steps, we calculate $DTW(\overleftarrow{\mathbf{x}}, \overleftarrow{\mathbf{h}})$. Then we evaluate possible back up step widths for the reverse time series $\overleftarrow{\mathbf{x}}$ and $\overleftarrow{\mathbf{h}}$ – which is equivalent to the catch up of the original time series \mathbf{x} and \mathbf{h} – by comparing the last row of $\overleftarrow{\mathbf{G}}$ analogously to the back up step of \mathbf{x} and \mathbf{h} . Further we can evaluate possible backward steps in parallel to the catch up step by calculating an incremental step of $\overleftarrow{\mathbf{G}}$ and comparing the last row. In Fig. 4d the Caterpillar algorithm decides to catch up and excludes the initial 5 observations of \mathbf{x} from alignment to \mathbf{h} and finally concludes with the optimal fit.

E. Complexity

The runtime complexity of the E-model and fDTW-model are identical to the calculation of the Euclidean distance $O(m)$, DTW distance $O(n \cdot m)$, respectively. One forward step of the Caterpillar algorithm has $O(n)$ complexity which is much less than computing it again from scratch at the costs of $O(n \cdot (m + 1))$. The decremental movement has $O((1 - 0.73) \cdot n)$ complexity according to (9). The same is valid for the reverse movements (except for the initial reverse movements).

V. RESULTS

To demonstrate that our proposed method detects time warped instances of a hypothesis time series \mathbf{h} , we compare (1) the proposed encoding scheme isolated from scanning algorithms with traditional distance based methods and similar state of the art encoding schemes for linear and non-linear alignments on synthetic data (Sections V-A and (2) the Caterpillar, PREMDL and UCR Suite [15] on synthetic data and accelerometer data (Sections V-B and V-C).

Our comparative evaluation includes two additional models from the literature which describe a time series’ DL and CDL based only on the cardinality c of a time series. The first model introduced by [2] defines $DL(\mathbf{x} | \mathbf{h})$ by counting the mismatches of \mathbf{x} and \mathbf{h} , and for each mismatch the costs for one observation plus the index costs are saved (we call it **C-Model**). The second additional model proposed by [21] is an

extension of the C-Model for time series of possibly different lengths by incorporating DTW and the coding scheme of [2]. They propose to count the mismatches of \mathbf{x} and \mathbf{h} along the warping path of these two time series (we call it **cDTW-Model**). Both models depend on the number of mismatches, however are independent of the magnitude of the mismatches.

Since the Caterpillar scans longer time series to detect shorter query patterns, we compare the Caterpillar against this task’s state-of-the-art method **UCR Suite** [15] (applying DTW or US). UCR Suite compares a given query pattern with segments of the same length of a longer time series via a sliding window approach. The DTW distance is calculated for each comparison (unless lower bounding avoids it), and early abandoning aborts the calculation. The original algorithm returns the segment with the lowest DTW distance and the distance itself. We apply UCR Suite multiple times to return multiple fits in time series (see Figures 1c and d). UCR Suite US is a combination of pruning and lower bounding techniques to accelerate calculations and a generalization of the Euclidean Distance – Uniform Scaling – such that fits of different lengths (we allowed scaling factors between 0.5 and 2) can be detected.

For runtime comparisons we used a standard laptop computer with 2.8 GHz and 16GB RAM.

A. Determine Matches in Synthetic Time Series

To investigate under which conditions our proposed model **fDTW** and others perform best, we test them independently from heuristic pattern matching algorithms, that is we only compare complete alignments of two time series \mathbf{x} and \mathbf{h} for models incorporating DTW and selected the lowest CDL for linear alignments (which is similar to PREMDL). With the help of the MDL framework we decide if two time series are either ‘not close’ or ‘close’ and assign them to each other. For traditional DTW and UCR Suite we need to set a threshold that separates ‘not close’ from ‘close’. We simulate couples of discretized $N(0,1)$ Markov Chain random walk time series (\mathbf{x}, \mathbf{h}) , where $x_0 = 0$, $x_i = x_{i-1} + z$ and $z \sim N(0, 1)$. The hypothesis \mathbf{h} is either independently simulated from \mathbf{x} , or simulated as a noisy and warped instance of \mathbf{x} . We varied the simulation parameters of $n = |\mathbf{h}|$, $\sigma =$ standard deviation of the overlaid noise and the amount of warp. In this synthetic case, we do not consider the costs of the hypothesis, since the models are supposed to be applied to find multiple matches to one hypothesis. The costs of each hypothesis would be compared to the total savings of all found fits per hypothesis.

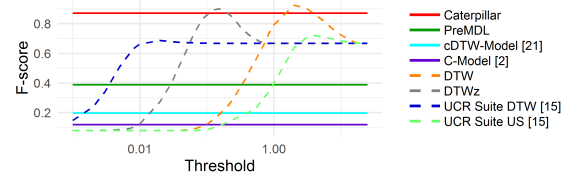


Figure 5: F-scores per model and threshold for detecting whether two time series are similar or not.

Figure 5 shows different models and the according F-scores of the classification task whether one x matches the according h or not. The Caterpillar (the fDTW-model respectively) performs almost as good as a classifier applying DTW and setting the best threshold (DTWz performs z-normalisation before the distance calculation). The low levels of the F1-scores of PREMDL and the cardinality based models – C-model [2] and cDTW-model [21] – are due to a low recall. The models UCR Suite DTW [15], PREMDL and C-Model cannot fit time series of different lengths, so they scan the longer time series for the best fit. Figure 6 plots the F-scores per

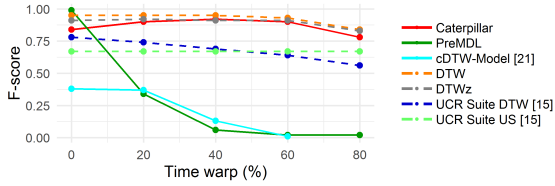


Figure 6: F-scores per model and degree of warping for complete alignments of synthetic time series

amount of time warp. For the methods requiring identifying a distance threshold we set the threshold equal to values showing best performance in Fig. 5. In case of no time warp, the PREMDL performs best, however already at a low level of 10% warping the performance drops from 96% to 72%. As expected, the models relying on DTW perform much better with F1-scores above 90%, however the cDTW-model cannot compete with DTW, Suite DTW and the Caterpillar. The C-Model does not identify any matches in case of warped time series, leading to F1-scores of NaN, due to division by 0. The Caterpillar performs almost as good as the basic DTW (with the best threshold set) with F1-scores between 90% and 96%. The performance of UCR Suite US is independent from the warping extend which is because the nature of US. The accepted percentage of time warp certainly depends on the application. As Figure 1 demonstrates a warping to the extent of 56% (= 28/50 sec) is reasonable.

B. Scan for multiple Matches in Synthetic Time Series

We simulate x as a concatenation of multiple warped and noisy instances of h and noise-like random walks. Since we know the points in time when the patterns start and end, we evaluate the techniques by counting mismatches and measuring the relative distances of detected fits to their ground truth instance. Figure. 7 and Fig. 8 show the performance and computation time. The Caterpillar (8.7% deviation and 6.7% misses) outperforms the other methods, where PREMDL is the second best (23.2% deviation and 18.5% misses).

Since UCR Suite is designed to find the (single) segment in a long time series best matching a given query pattern, we need to invoke it multiple times to find multiple fits. It must be stressed that UCR Suite (DTW and US) could be much faster for finding multiple fits if it were adapted to detect the k nearest segments instead of one, or alternatively all fits beyond a predefined *threshold*. However, such an adaption still would

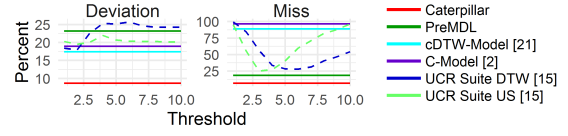


Figure 7: Errors for finding multiple fits in synthetic data

leave the open question of identifying suitable parameters. In addition, while the computation time would be lower, the found fits would be the same. We leave it for future work to test computation times of such a non-trivial adapted version of UCR Suite. Caterpillar is the most accurate method for accurately detecting multiple fits and their extensions. The cDTW-Model is the second best method in terms of deviation, but misses many matches. As for PreMDL, the opposite holds. Our algorithm clearly outperforms both methods at a minor runtime overhead. For comparison we implemented the brute force method of UCR Suite US [15], so the original algorithm applying enveloping techniques would be even faster, however the error rates are worse than most other methods here.

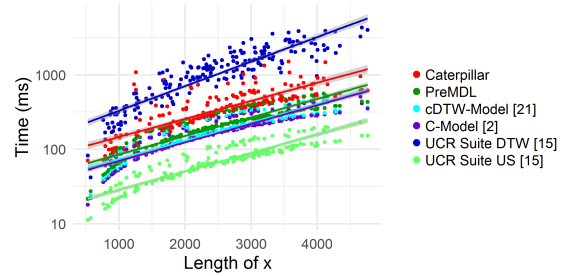


Figure 8: Run times for finding multiple fits in synthetic data.

C. Smartphone Accelerometer Data

To demonstrate the broad applicability of the Caterpillar we use it to detect transport mode specific patterns. We use data collected in an urban area by ten participants equipped with a mobile application on their private smartphones to collect acceleration sensor data. In total the participants collected about 50 hours of unimodal trips of different transport modes (bicycle 7 hours, bus 1, car 3, metro 14, train 4, tram 7, walk 15) and annotated their trips live via the app. For preprocessing, we extracted the horizontal and vertical component of the collected signal similar to [6], [12], [14] and concentrate further only on the horizontal signal. Figure 1b depicts the hypothesis signal h we extracted from a scripted trip that represents a typical pattern of a metro that brakes, stops and accelerates again. As Figure 10 shows, such patterns vary vertically and in time extension. Depending on the circumstances (the metro driver, the distance between two stations, the number of passengers, etc.), the metro’s acceleration and braking pattern may last longer or shorter. Also, metros do not stop for exactly the same time. Our proposed method still finds these patterns in the acceleration time series.

To evaluate our proposed method, we need a setup with known ground truth. A trip is classified as a metro trip

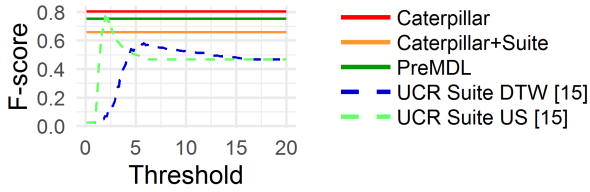


Figure 9: F-scores for classifying accelerometer time series as metro or non-metro

if it contains at least one fit to the hypothesis. Figure 9 demonstrates that the Caterpillar performs best in detecting these metro-typical patterns and achieves an F-score of 80%, which is 5% better than PREMDL and 22% better than UCR Suite DTW (setting the best threshold). Especially UCR Suite DTW has a high recall of 96%, and a low precision of 41%. Setting the ideal threshold UCR Suite US is almost as good as the Caterpillar with an F1-score of 77%. The performance of the UCR Suite methods (dashed lines) highly depends on setting the right threshold. We used UCR Suite in its original form to detect single fits and in the described adapted form to detect multiple fits (see Sec. V-B), both on the original data and discretized time series. The F1-scores for these four variations vary for about 2%, and Fig. 9 shows the best, the multiple Suite DTW on the discrete data with an F1-score of 58%. We also used Suite to detect initials for the Caterpillar (Caterpillar+Suite in Fig. 9), and achieved an F1-score of 66%. We also used (not plotted) only those fits found by Suite below the threshold that achieves the best performance for Suite itself (5.8), and achieved an F1-score of 62%.

To illustrate the dilemma of setting a valid threshold, Figure 10 zooms in the first few minutes of the trip data introduced in Fig. 1. The initial grey segment in Fig. 10d with a US distance of 5 to the query pattern resembles the query pattern obviously much better than the following three segments with US distances of 2, 4, and 3 which show rather random patterns. The two light blue segments starting at seconds 200 and 280 show low distances and visually reasonable matches. Also for the fits found by DTW in Fig. 10c, the distances show a counterintuitive picture. The purple segment at second 280 has a DTW distance of 8 and resembles the query pattern better than the purple segment at second 40 (distance 8) or the light blue second 70 (distance 4). We conclude that it is difficult to set a threshold that separates good from bad fits. The Caterpillar fuses DTW and the MDL principle to avoid such sensitive parameters and performs much better on these data. Moreover the Caterpillar is capable to detect fits of different lengths, whereas those of PREMDL and UCR Suite DTW all have the same length as the query pattern. UCR Suite US also detects fits of different lengths, and performs almost as good as the Caterpillar on the accelerometer data, but as Sec. V-B shows worse on random walk-like synthetic data.

VI. RELATED WORK AND DISCUSSION

The MDL principle has been used for controlling the model complexity in different areas (e.g. bioinformatics [20], text

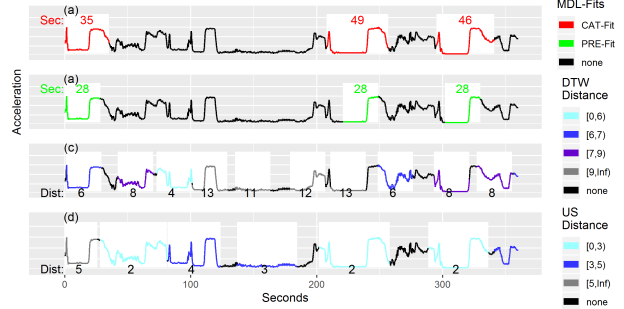


Figure 10: Zoomed version of Fig. 1. Detected fits of Caterpillar (a), PREMDL (b), UCR Suite DTW (c) and UCR Suite US (d)

mining [5], etc.) In time series analysis [2], [21], [7], [16], [22], the MDL principle is useful to answer the question ‘Is a given time series x close enough to another h to consider them as a match?’. To answer this question, the DL of one time series is compared to its CDL given another time series (as applied in Section V-A). In general two strategies exist in the literature to formulate encoding models for the DL and CDL based on MDL. The first approach is to use the cardinality of time series to model the DL [2], [21], [7]. Here the CDL only depends on the number of deviations of two time series, but not the magnitude of the deviations. The second approach is to model the DL and CDL via entropy with Huffman coding [16], [22]. In this work we combine both approaches and demonstrate our method to outperform similar approaches and to be widely applicable.

DTW has been used in many works for time series analysis [3], [9], [4]. There are different approaches how to speed up or early abandon the DTW calculation ([9], [1], [18], [10], [19]) which are all combinable with the here proposed fDTW-model for complete matches of time series (as in Sec. V-A). We leave it for future work to investigate if the Caterpillar algorithm could further benefit from early abandoning or lower bounding.

One of the most relevant works in the field of time series matching is the presentation of UCR Suite [15], that facilitates fast comparisons of many time series or scanning of longer time series for query patterns. However UCR Suite DTW only detects matches of the same length as the query pattern, in contrast to our proposed method, the Caterpillar algorithm, and UCR Suite US that we also benchmark against. Due to selecting initials at the cost of $O(n)$, incremental DTW calculation and recycling of former results we save computation time. The Caterpillar efficiently and reliably detects multiple warped fits in a long time series. Relying on the MDL Principle our algorithm automatically fulfils this task without requiring the user to specify input parameters which are difficult to set.

The work in [1] applies the incremental calculation of DTW and also proves it to be reversible to use it for anticipatory pruning. We use these insights and complete the ways of piecewise DTW calculations for alterations of past observations: reverse in-/decremental calculations.

In contrast to [8], [16], [15] we normalize and discretize time series as a whole (see 2), because of the different nature of methods relying on sliding windows of equal lengths and the here proposed incremental comparison of segments.

More specifically, for an incremental step of the Caterpillar, the parameters for normalization (mean, standard deviation, min, max) either would need to be redefined for \mathbf{x}^+ , but normalizing \mathbf{x}^+ with different parameters than \mathbf{x} implicates that former results of the DTW calculations can not be recycled, and DTW (or the fDTW-model) needs to be recalculated from scratch. This would make the problem of finding multiple fits of different lengths almost unsolvable in reasonable time. On the other, hand if the parameters are reused for \mathbf{x}^+ , we could run out of range: e.g. let $\mathbf{h}=(2,2,2,4)$ and $\mathbf{x}=(1,1,1,2,4)$, and we start with an initial warping of \mathbf{h} and the first four elements of \mathbf{x} , $\mathbf{x}_{1:4}$. For the incremental step we test to align the last element of \mathbf{x} , 4. If we normalize $\mathbf{x}_{1:5}$ according to (2) with the same parameters as $\mathbf{x}_{1:4}$, the cost matrix of DTW has entries beyond the range of P , and cannot be encoded with the here proposed encoding scheme. We leave it for future work to investigate possible adjustments of the Caterpillar with running normalization for streaming data that follows no physical constraints (as e.g. accelerometer records of smartphones).

Further we demonstrate to outperform UCR Suite, relying on segmentwise normalization, on accelerometer data and synthetic data. The Caterpillar also allows vertical shifts and can compensate a wandering baseline, such that the two patterns (1,2,3,3,2,4) and (11,12,13,13,12,14) have the same CDL to a query pattern (2,3,4,4,3,5), due to (7).

VII. CONCLUSION AND OUTLOOK

In this work we presented a novel encoding scheme for time series applying the MDL framework. We compared the proposed model against state of the art methods on synthetic data and further we applied the model for our proposed Caterpillar algorithm that enables to identify vertically shifted and time warped matches of different lengths of hypothesis time series. Finally we demonstrated the Caterpillar to identify metro stops in accelerometer time series. For future work we plan to investigate the potential of DTW pruning methods for the Caterpillar algorithm and test adjustments of the Caterpillar for streaming data as e.g. financial time series.

REFERENCES

- [1] I. Assent, M. Wichterich, R. Krieger, H. Kremer, and T. Seidl. Anticipatory dtw for efficient similarity search in time series databases. *Proc. VLDB Endow.*, 2(1):826–837, Aug. 2009.
- [2] N. Begum, B. Hu, T. Rakthanmanon, and E. Keogh. Towards a minimum description length based stopping criterion for semi-supervised time series classification. In *2013 IEEE 14th International Conference on Information Reuse Integration (IRI)*, pages 333–340, Aug 2013.
- [3] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, AAAIWS'94*, pages 359–370. AAAI Press, 1994.
- [4] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.
- [5] P. D. Grnwald, I. J. Myung, and M. A. Pitt. *Advances in Minimum Description Length: Theory and Applications (Neural Information Processing)*. The MIT Press, 2005.
- [6] S. Hemminki, P. Nurmi, and S. Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, SenSys '13*, pages 13:1–13:14, New York, NY, USA, 2013. ACM.
- [7] B. Hu, T. Rakthanmanon, Y. Hao, S. Evans, S. Lonardi, and E. Keogh. Discovering the intrinsic cardinality and dimensionality of time series using mdl. In *2011 IEEE 11th International Conference on Data Mining*, pages 1086–1091, Dec 2011.
- [8] E. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems*, 8(2):154–177, 2005.
- [9] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, Mar 2005.
- [10] E. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos. Lb_keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *Proceedings of the 32nd international conference on Very large data bases*, pages 882–893. VLDB Endowment, 2006.
- [11] M. Leodolter. R-package for incremental dynamic time warping, July 2017. <https://cran.r-project.org/web/packages/incDTW/index.html>.
- [12] M. Leodolter, P. Widhalm, C. Plant, and N. Brandle. Semi-supervised segmentation of accelerometer time series for transport mode classification. In *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 663–668, June 2017.
- [13] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03*, pages 2–11, New York, NY, USA, 2003. ACM.
- [14] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, SenSys '10*, pages 71–84, New York, NY, USA, 2010. ACM.
- [15] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(3):10, 2013.
- [16] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans. Mdl-based time series clustering. *Knowledge and Information Systems*, 33(2):371–399, 2012.
- [17] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, Feb 1978.
- [18] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. Ftw: fast similarity search under the time warping distance. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 326–337. ACM, 2005.
- [19] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.
- [20] W. Szpankowski, W. Ren, and L. Szpankowski. An optimal dna segmentation based on the mdl principle. *International Journal of Bioinformatics Research and Applications*, 1(1):3–17, 2005. PMID: 18048118.
- [21] V. Thanh Vinh and D. Tuan Anh. *Some Novel Improvements for MDL-Based Semi-supervised Classification of Time Series*, pages 483–493. Springer International Publishing, Cham, 2014.
- [22] V. T. Vinh and D. T. Anh. Constraint-based mdl principle for semi-supervised classification of time series. In *2015 Seventh International Conference on Knowledge and Systems Engineering (KSE)*, pages 43–48, Oct 2015.

Paper C

IncDTW: An R Package for Incremental Calculation of Dynamic Time Warping

License Information

The Journal of Statistical Software has chosen to apply the Creative Commons Attribution License to all articles published in its journal. Therefore, this paper is licensed under the Creative Commons Attribution License¹. The original work was not changed for presentation in this thesis.

¹<https://creativecommons.org/licenses/by/3.0/>



Journal of Statistical Software


September 2021, Volume 99, Issue 9.

doi: 10.18637/jss.v099.i09

IncDTW: An R Package for Incremental Calculation of Dynamic Time Warping

Maximilian Leodolter 
Austrian Institute
of Technology

Claudia Plant 
University of Vienna

Norbert Brändle 
Austrian Institute
of Technology

Abstract

Dynamic time warping (DTW) is a popular distance measure for time series analysis and has been applied in many research domains. This paper proposes the R package **IncDTW** for the incremental calculation of DTW, and based on this principle **IncDTW** also helps to classify or cluster time series, or perform subsequence matching and k -nearest neighbor search. DTW can measure dissimilarity between two temporal sequences which may vary in speed, with a major downside of high computational costs. Especially for analyzing live data streams, subsequence matching or calculating pairwise distance matrices, runtime intensive computations are unfavorable or can even make the analysis intractable. **IncDTW** tackles this problem by a vector-based implementation of the DTW algorithm to reduce the space complexity from a quadratic to a linear level in number of observations, and an incremental calculation of DTW for updating interim results to reduce the runtime complexity for online applications.

We discuss the fundamental functionalities of **IncDTW** and apply the package to classify multivariate live stream accelerometer time series for activity recognition. Finally, comparative runtime experiments with various R and Python packages for various data analysis tasks emphasize the broad applicability of **IncDTW**.

Keywords: dynamic time warping, time series, k -NN, subsequence matching, distance measure, clustering, classification.

1. Introduction

Time series are sets of observations that follow a consecutive temporal relation. Many time series data analysis tasks such as clustering, classification, outlier detection or pattern matching require the definition of a distance measure. Many distance measures such as the Euclidean distance are rather ill-suited whenever two time series are shifted in time, locally recorded with different sampling rates, warped, or have different lengths. Dynamic time warping (DTW)

was originally proposed by Sakoe and Chiba (1978), and has since been the distance measure of choice in many works for time series analysis (Berndt and Clifford 1994; Keogh 2002; Ding, Trajcevski, Scheuermann, Wang, and Keogh 2008; Kwankhoom and Muneesawang 2017; Oregi, Pérez, Del Ser, and Lozano 2017; Giorgino *et al.* 2009). DTW is capable of dealing with deformed time series by identifying the best alignment of two time series in a dynamic way.

The major downside of DTW are its expensive computational costs, which are particularly unfavorable for online algorithms processing continuous data streams, where time series analysis must be faster than the elapsed time between consecutive observations. One solution to reduce the runtime for online processing is to incrementally calculate DTW by recycling interim results of previous calculations for every new observation. Without any loss of accuracy, such an incremental processing allows reducing computation time complexity towards linear level in number of observations. Section 2.1 and 3.2 give a detailed discussion about the runtime and space complexity of the DTW algorithm.

The groundwork for the incremental calculation of DTW was done by Rabiner, Rosenberg, and Levinson (1978), who proposed adjustments to the DTW algorithm - open alignments. Since then the principle of the incremental DTW computation has been applied in multiple works, as e.g.: Dixon (2005) applied it for an online algorithm to track musical performances, Mori, Uchida, Kurazume, Taniguchi, Hasegawa, and Sakoe (2006) for an algorithm to early recognize gestures, Tormene, Giorgino, Quaglini, and Stefanelli (2008) to analyze multivariate sensor readings to support neurological patients with real-time information while undergoing motor rehabilitation, Kwankhoom and Muneesawang (2017) for online algorithms which re-identify movement trajectories of persons captured with a 3D depth sensing camera, where time series matching is updated as soon as new video frames are recorded, and Oregi *et al.* (2017) for proposing the Online-DTW (ODTW) algorithm.

Dynamic time warping has already been applied in many research domains and also published in different software packages and programming languages. Table 1 gives an overview of R (R Core Team 2021) packages for DTW computation available at the Comprehensive R Archive Network (CRAN) at <https://CRAN.R-project.org/>, and Python (Van Rossum *et al.* 2011) packages available at the Python Package Index (PyPI) at <https://pypi.org/>. The package `dtw` (Giorgino *et al.* 2009) offers functions for DTW calculation with different step patterns (see (2) and (3)), warping path restrictions and plotting functions, also for a profound visual analysis of warping alignments of two time series. `dtwclust` (Sarda-Espinosa 2019) puts emphasis on clustering time series based on DTW distances. The functions for DTW calculations are wrappers for those of the `dtw` package. The package `dtwSat` (Maus *et al.* 2019) provides with the time-weighted dynamic time warping a distance method customized to analyzing satellite image time series. `rucrdtw` (Boersch-Supan 2016) is the R version of UCR Suite (Rakthanmanon, Campana, Mueen, Batista, Westover, Zhu, Zakaria, and Keogh 2012) which is a nearest neighbor search algorithm accelerated by lower bounding and pruning methods. It detects the closest fit to a query time series in either one long time series or many of the same length. To the best of our knowledge `rucrdtw` is – besides `IncDTW` – the only R package with a vector-based implementation of the DTW algorithm, thus avoiding memory allocation of matrices. However, the package does neither support multivariate time series nor full alignments for time series of different lengths (i.e., from begin to end for both time series). The package `parallelDist` (Eckert 2018) is the parallel implementation of the function `dist()` – of the package `stats` (R Core Team 2021) – by incorporating

Package & Repo.	First release	Incremental	Vector-based	Diff. lengths	Multi-variate	k -NN	Description / Focus
IncDTW CRAN	2017	Yes	Yes	Yes	Yes	Yes	Incremental and fast vector-based DTW calculations, described in this paper.
dtw CRAN	2007	No	No	Yes	Yes	No	Highly functional implementation of DTW (Giorgino <i>et al.</i> 2009).
dtwclust CRAN	2015	No	No	Yes	Yes	No	Time series clustering with DTW (Sarda-Espinosa 2019).
dtwSat CRAN	2015	No	No	Yes	Yes	No	Time-Weighted DTW for satellite images (Maus, Câmara, Appel, and Pebesma 2019).
rucrdtw CRAN	2016	No	Yes	No	No	No	1NN-search via DTW (Boersch-Supan 2016).
parallelDist CRAN	2017	No	No	Yes	Yes	No	Parallel distance calculation (Eckert 2018).
dtw PyPI	2014	No	No	Yes	Yes	No	Highly forked and starred (Rouanet 2014).
dtaidistance PyPI	2017	No	Yes	Yes	No	No	Functional and fast (Meert 2017).
cydtw PyPI	2017	No	No	Yes	Yes	No	Simple and fast (Tavenard 2017).

Table 1: Overview of various R and Python packages with different emphasis on calculating and applying the DTW distance.

RcppParallel (Allaire, François, Ushey, Vandenbrouck, Geelnard, and Intel 2021) to speed up computations. Apart from R packages for DTW computation, in the following we discuss Python software, since, – similar to R – Python is probably one of the most taught and applied programming languages for time series analysis, and data mining tasks as clustering, classification and pattern recognition applied on time series data. The **dtw** package (Rouanet 2014) is one of the highest forked and starred packages for DTW computation on the Python Package Index (<https://pypi.org/>). To compute the DTW distance for multivariate time series the package **cydtw** (Tavenard 2017) offers a solution, and the main computation part of the algorithm is implemented in C via **Cython** (Behnel, Bradshaw, Citro, Dalcin, Seljebotn, and Smith 2011). The package **dtaidistance** (Meert 2017) offers a more comprehensive set of functions and also a vector based implementation in C via **Cython**, but does not support multivariate time series. Next to the R packages, Table 1 also lists the Python packages and Section 4.2 details a runtime experiment which compares the Python packages with **IncDTW**. The main contributions of this paper and the R package **IncDTW** are:

- the principle of the incremental DTW calculation ready to use in R functions,
- vector-based implementation of the DTW algorithm – also for multivariate time series – to decrease the computation time,
- the demonstration of applying **IncDTW** for an online time series classification task,
- and comparative runtime experiments with other R and Python packages.

This paper is organized as follows: Section 2 gives an introduction to DTW in general and explains the incremental calculation. Section 3 describes the R package **IncDTW**, discusses the vector-based functions and how to apply the incremental calculation. Section 4 discusses the advantages of **IncDTW** by hand of a typical time series classification experiment, and shows runtime comparisons. Section 5 concludes this paper and gives an outlook of future developments.

2. Dynamic time warping

In the following we recapitulate the classic dynamic time warping algorithm from [Sakoe and Chiba \(1978\)](#) which calculates the distance measure between a query time series \mathbf{q} and a candidate time series \mathbf{c} , and their alignment – the so-called warping path – providing information which observations of \mathbf{q} are best matched to the respective observations of \mathbf{c} .

The distance measure DTW is defined as the minimal cumulative costs of the shortest non-linear alignment of two time series \mathbf{q} and \mathbf{c} . This alignment has the following properties:

1. **Boundary conditions:** The first element of \mathbf{q} is aligned to the first element of \mathbf{c} , and the last element of \mathbf{q} is aligned to the last element of \mathbf{c} . Relaxing these conditions allows to find an open alignment, i.e., a partial alignment of two time series with lowest DTW distance (normalized for the lengths). For a more detailed discussion on open alignments (open-end, open-begin and open increment) we refer to the package vignette of **IncDTW** ([Leodolter 2021](#)).
2. **Monotonicity:** Consecutive elements of \mathbf{q} and \mathbf{c} must not be aligned out of time order. The DTW algorithm also returns vectors of indices of \mathbf{q} and \mathbf{c} defining the ordering of the best aligned observations. These vectors must be monotonically increasing, such that $i_k \leq i_{k+1}$, where $1 \leq i_k \leq n = |\mathbf{q}|$, and i_k defines which elements of \mathbf{q} are aligned to \mathbf{c} at the k -th point of time. The same applies to the indices $j_k \leq j_{k+1}$ defining which elements of \mathbf{c} are aligned to \mathbf{q} at the k -th point of time.
3. **Non-linear alignment:** In contrast to the Euclidean distance, one observation of \mathbf{q} can be aligned to more than one observation of \mathbf{c} , and vice versa. Hence it is possible that $i_k = i_{k+1}$ or $j_k = j_{k+1}$.
4. **Restrictions:** Global or local warping path restrictions can be applied to reduce the space of possible alignments. The most known is the Sakoe Chiba warping window ([Sakoe and Chiba 1978](#)), where the time difference of two aligned observations must not exceed the window size parameter, ω : $|i_k - j_k| \leq \omega \forall k$.

5. Local distance measure: The distance of two (possibly multivariate) observations of the time series \mathbf{q} and \mathbf{c} can be defined by any distance metric. The standard metrics are the 1-norm and 2-norm. The package vignette of **IncDTW** elaborates how to customize the local distance functions.
6. Step Pattern: The step pattern defines how the local distances are accumulated to calculate the global cost matrix and the walking path. The two popular step patterns (2) and (3) are implemented in **IncDTW**. Sakoe and Chiba (1978), Rabiner and Juang (1993) or Giorgino *et al.* (2009) give a more detailed discussion on step patterns.

It is worth noting that the DTW distance measure is not a metric, since it does not fulfill the triangle inequality. Consequently, lower bounding with the help of the reverse triangle inequality is not possible, which is a method applied for fast nearest neighbor search (Wang 2011).

For the two time series \mathbf{q} of length n and \mathbf{c} of length m , we define $\mathbf{C} \in \mathbf{R}^{n \times m}$ as the local cost matrix, where

$$\mathbf{C}_{i,j} := d(\mathbf{q}_i, \mathbf{c}_j), \quad (1)$$

with $d(\cdot, \cdot)$ as a local distance function for univariate or multivariate time series as described above. The global cost matrix $\mathbf{G} \in \mathbf{R}^{n \times m}$ is determined in an iterative fashion, where each element depends on its predecessors. The step pattern defines these dependencies by weighting and selecting the predecessors. Giorgino *et al.* (2009) and Rabiner and Juang (1993) present a more detailed discussion on step patterns, here we concentrate on two of the most popular and start with the naive step pattern that regards the direct neighboring elements in \mathbf{G} equally weighted:

$$\mathbf{G}_{i,j} = \begin{cases} \sum_{k \leq i} \mathbf{C}_{k,1} & j = 1 \\ \sum_{l \leq j} \mathbf{C}_{1,l} & i = 1 \\ \mathbf{C}_{i,j} + \min(\mathbf{G}_{i-1,j}, \mathbf{G}_{i,j-1}, \mathbf{G}_{i-1,j-1}) & i, j > 1. \end{cases} \quad (2)$$

The step pattern described by Algorithm 2 – “symmetric1” – was not part of the original work about DTW of Sakoe and Chiba (1978) since it does not admit a normalization factor. Nevertheless, it has been applied in several works (Fu 2011; Berndt and Clifford 1994; Sakurai, Faloutsos, and Yamamuro 2007; Keogh 2002; Rath and Manmatha 2003b; Keogh and Pazzani 2000; Rakthanmanon *et al.* 2012) about time series clustering, classification, indexing and pattern mining, and so gained popularity, possibly due to its simplicity to understand and implement.

Another typical step pattern, that is also the default step pattern in the R package **dtw**, is called “symmetric2”. Here the diagonal step is weighted with a weight of 2:

$$\mathbf{G}_{i,j} = \begin{cases} \sum_{k \leq i} \mathbf{C}_{k,1} & j = 1 \\ \sum_{l \leq j} \mathbf{C}_{1,l} & i = 1 \\ \min(\mathbf{C}_{i,j} + \mathbf{G}_{i-1,j}, \mathbf{C}_{i,j} + \mathbf{G}_{i,j-1}, 2 \cdot \mathbf{C}_{i,j} + \mathbf{G}_{i-1,j-1}) & i, j > 1. \end{cases} \quad (3)$$

The step pattern (3) is also discussed as special case of the general formulation in Sakoe and Chiba (1978). The direction matrix $\mathbf{D} \in \mathbf{N}^{n \times m}$ gives information about the alignment of

Algorithm 1 Backtracking the direction matrix \mathbf{D} delivers the warping path \mathbf{w} .

```

1: procedure BACKTRACKING( $\mathbf{D}$ )
2:    $i \leftarrow n$   $\triangleright n = \text{length of the time series } \mathbf{q}$ 
3:    $j \leftarrow m$   $\triangleright m = \text{length of the time series } \mathbf{c}$ 
4:    $\mathbf{w}, \mathbf{ii}, \mathbf{jj} \leftarrow$  empty vectors
5:   repeat
6:      $\text{step} \leftarrow \mathbf{D}(i, j)$ ;
7:     if  $\text{step} == 1$  then
8:        $i \leftarrow i - 1$ 
9:        $j \leftarrow j - 1$ 
10:    else if  $\text{step} == 2$  then
11:       $j \leftarrow j - 1$ 
12:    else
13:       $i \leftarrow i - 1$ 
14:    end if
15:     $\mathbf{ii} \leftarrow \text{append}(i, \mathbf{ii})$ 
16:     $\mathbf{jj} \leftarrow \text{append}(j, \mathbf{jj})$ 
17:     $\mathbf{w} \leftarrow \text{append}(\text{step}, \mathbf{w})$ 
18:  until  $i < 0 \mid j < 0$  return  $\mathbf{w}, \mathbf{ii}$  and  $\mathbf{jj}$ 
19: end procedure

```

the two time series and is calculated simultaneously with the calculation of \mathbf{G} . The following equation defines \mathbf{D} for the step pattern of (2):

$$\mathbf{D}_{i,j} = \begin{cases} 1 & \text{if } \mathbf{G}_{i,j} = \mathbf{C}_{i,j} + \mathbf{G}_{i-1,j-1} \\ 2 & \text{if } \mathbf{G}_{i,j} = \mathbf{C}_{i,j} + \mathbf{G}_{i,j-1} \\ 3 & \text{if } \mathbf{G}_{i,j} = \mathbf{C}_{i,j} + \mathbf{G}_{i-1,j}. \end{cases} \quad (4)$$

The DTW distance measure is stored in the last column of the last row of \mathbf{G} , \mathbf{G}_{nm} , and indicates the cheapest cumulative costs to align \mathbf{q} and \mathbf{c} . The warping path vector \mathbf{w} is an excerpt of the direction matrix \mathbf{D} and achieved by backtracking \mathbf{D} . Starting at the last row and last column of \mathbf{D} , backtracking (Algorithm 1) checks the cheapest next step (1 is diagonal, 2 is horizontal, 3 is vertical) and stores this integer in a vector. The backtracking algorithm also returns the vectors \mathbf{ii} and \mathbf{jj} , the vectors of indices of \mathbf{q} and \mathbf{c} for the best alignment in the respective order.

2.1. Incremental DTW calculation

Calculating the DTW distance measure is computationally expensive, especially for long time series without a warping window, due to the quadratic runtime complexity $O(n \cdot m)$, where n and m are the lengths of the time series \mathbf{q} and \mathbf{c} , respectively. If the DTW distance is calculated with a Sakoe Chiba warping window of size ω , where $|m - n| \leq \omega \leq \max(m, n)$, the runtime complexity reduces to $O(\omega \cdot \min(m, n))$. Consequently, if ω increases and approaches its maximum value, then the runtime complexity approximates the quadratic level, and conversely if ω decreases, then it approximates a linear level. The space complexity is discussed in Section 3.2.

To update an alignment of two time series after recording new observations, it is possible to reuse interim results instead of calculating DTW from scratch. So, if a time series \mathbf{c} is observed for $i = 1 \dots m$, calculating the DTW distance measure to \mathbf{q} of length n has a runtime complexity¹ of $O(n \cdot m)$. As soon as new observations of \mathbf{c} are recorded for $i = m+1 \dots m+k$, calculating the DTW distance measure from scratch has a runtime complexity of $O(n \cdot (m+k))$. Contrary the incremental approach is based on storing the necessary components of the results of the initial DTW computation after observing \mathbf{c} for $i = 1 \dots m$, and recycling these interim results when new observations are recorded. This way the incremental update of the DTW distance at time $i = m+k$ has a runtime complexity of $O(n \cdot k)$. The examples in Section 3.3 and the experiment in Section 4.1 demonstrate this principle in more detail.

The input to incrementally calculate DTW of $\mathbf{q}[1 : n]$ and $\mathbf{c}[1 : m+k]$ is the output of the former calculation $\text{DTW}(\mathbf{q}[1 : n], \mathbf{c}[1 : m])$. This output is composed of three matrices: the global cost matrix \mathbf{G}^0 , the local cost matrix \mathbf{C}^0 and the direction matrix \mathbf{D}^0 . Additional required input is the time series of new observations of \mathbf{c} . To calculate the global cost matrix \mathbf{G}^1 of $\text{DTW}(\mathbf{q}[1 : n], \mathbf{c}[1 : m+k])$, we append new costs and direction entries to the previously calculated matrices and proceed analogously to (2):

1. First we build the local cost Matrix \mathbf{C}^1 :

$$\mathbf{C}_{ij}^1 := \begin{cases} \mathbf{C}_{ij}^0 & i \leq m \\ \text{dist}(q_i, c_j) & m < i \leq m+k. \end{cases} \quad (5)$$

2. Next the global cost matrix is appended to the former results and new entries are defined analogously to (2):

$$\mathbf{G}_{ij}^1 := \begin{cases} \mathbf{G}_{ij}^0 & i \leq m \\ \sum_{k \leq i} \mathbf{C}_{k,1} & j = 1 \\ \mathbf{C}_{i,j} + \min(\mathbf{G}_{i-1,j}, \mathbf{G}_{i,j-1}, \mathbf{G}_{i-1,j-1}) & \text{else} \end{cases} \quad (6)$$

3. The direction matrix \mathbf{D}^1 is calculated simultaneously to \mathbf{G}^1 .
4. Finally, the warping path needs to be calculated completely new from scratch, since in general it can not be excluded that new observations may open up completely different options to warp the two time series.

Equation 6 is the incremental version of (2). For (3) the definition of the new entries of \mathbf{G} is analogous, as for any other step pattern presented in Sakoe and Chiba (1978).

In fact, not the complete matrix \mathbf{G}^0 is required to update the DTW distance for new observations. Section 3.2 and 3.3 discuss an vector-based implementation for the incremental calculation that only requires the very last column (and row) of \mathbf{G}^0 .

Especially for live streaming data computation time is key. **IncDTW** (in particular the functions `increment()`, `idtw2vec()` and `idtw()` as demonstrated in Section 2 and 4) facilitates a fast update of time series distance measures when new observations arise. This can be of interest for any system analyzing live data streams.

¹For simplicity we reduce the following runtime complexity discussion for the general case of DTW calculation without a warping window. The derivation for DTW calculation with a warping window follows analogous arguments.

3. The R package IncDTW

This section describes the functions of the R package **IncDTW** and how to apply them to calculate the DTW distance: (1) Matrix-based, (2) vector-based, and (3) from scratch or incrementally. For details about further functionalities of **IncDTW** (e.g., an algorithm for searching the k -nearest subsequences of a time series with DTW, or time series clustering with DTW) we refer to the package documentation and vignette (Leodolter 2021). All results presented in this paper are achieved with version 1.1.4.3 of **IncDTW**. The computationally expensive parts of **IncDTW** are outsourced to C++ via the packages **Rcpp** (Eddelbuettel and François 2011) and **RcppArmadillo** (Eddelbuettel and Sanderson 2014), and parallelized via the packages **parallel** (R Core Team 2021) and **RcppParallel** (Allaire *et al.* 2021).

3.1. Matrix-based implementation

The classical DTW implementation relies on the local cost matrix \mathbf{C} , the direction matrix \mathbf{D} and the global cost matrix \mathbf{G} (see Section 2). \mathbf{C} can be stored as matrix or calculated entry-wise when \mathbf{G} is calculated. Returning the matrices \mathbf{G} and \mathbf{D} facilitates a detailed analysis of the alignment of two time series. Plotting Figure 2 for visual analysis is possible due to the information provided by the warping path, which in turn is an excerpt of the direction matrix \mathbf{D} and is achieved by backtracking. The entry \mathbf{C}_{ij} is the distance between q_i and c_j and can be described by any distance metric for univariate or multivariate time series dependent on the dimension of \mathbf{q} and \mathbf{c} . In case of multivariate time series, they need to have the same dimension, but still can vary in number of observations. In the univariate case the 1-norm is equivalent to the 2-norm, which is the absolute value of the difference $|q_i - c_j|$.

The basic DTW algorithm for computing the global cost matrix \mathbf{G} , according to (2), steps through the local cost matrix \mathbf{C} . The following parameters characterize in detail how the algorithm defines \mathbf{G} and finds the warping path:

- **dist_method**: The local distances are stored in \mathbf{C} , where $\mathbf{C}_{ij} = \text{dist_method}(q_i, c_j)$. So the parameter **dist_method** defines how the local distance of observations are measured. For O -dimensional time series the distances “norm1”, “norm2” and “norm2_square” are defined as:

$$\begin{aligned} \|\mathbf{q}_i, \mathbf{c}_j\|_1 &:= \sum_{o=1}^O |q_{io} - c_{jo}| \\ \|\mathbf{q}_i, \mathbf{c}_j\|_2 &:= \sqrt{\sum_{o=1}^O (q_{io} - c_{jo})^2} \\ \|\mathbf{q}_i, \mathbf{c}_j\|_2^2 &:= \sum_{o=1}^O (q_{io} - c_{jo})^2. \end{aligned} \tag{7}$$

Apart from these three predefined local distance functions **IncDTW** also allows to define customized local distance functions.

- **ws**: The space of all possible alignments of two time series can be constrained by warping windows. As Section 2 mentions, the most popular constraint is the Sakoe-Chiba window (Sakoe and Chiba 1978), which adjusts the DTW algorithm by setting $\mathbf{G}_{ij} = \infty$

if $|i - j| > \mathbf{ws}$. So \mathbf{ws} defines the window size of allowed warping paths. If we set $\mathbf{ws} = 0$ then only the diagonal of \mathbf{G} is used for aligning \mathbf{q} and \mathbf{c} , which is identical to the Euclidean distance. In this case the time series must have the same length. If the lengths of the time series differ more than \mathbf{ws} , then obviously no valid alignment can be found.

- **step_pattern**: The step pattern defines how the DTW algorithm finds the cheapest path through the local cost matrix. In (2) the most basic and broadly applied step pattern “symmetric1” is used, where the direct neighbors are considered and all are weighted equally. In (3) the step pattern “symmetric2” is uses a weight of 2 for the diagonal step and 1 for the vertical and horizontal to compensate the favor of diagonal steps. The current version of **IncDTW** concentrates on these two patterns and we consider other step patterns for future developments. For a more detailed discussion of step patterns we refer to [Giorgino *et al.* \(2009\)](#) and [Rabiner and Juang \(1993\)](#).

The following commands install and load the package **IncDTW**:

```
R> install.packages("IncDTW")
R> library("IncDTW")
```

First we define the help function `rw()` (which we also use in the next sections) to simulate a Gaussian random walk. Then a basic calculation of the DTW distance is done as follows:

```
R> rw <- function(n) cumsum(rnorm(n))
R> Q <- rw(100)
R> C <- rw(80)
R> result <- dtw(Q, C, ws = 30, step_pattern = "symmetric2")
R> result$distance
```

```
[1] 197.1266
```

3.2. Vector-based implementation

The matrix-based implementation is necessary for a detailed analysis of the alignment of two time series since it allows to calculate and return the warping path. Tasks such as nearest neighbor search, or the calculation of a matrix of pairwise distances to cluster or classify a database of time series require many DTW computations, and so the computation time of DTW is a major bottleneck.

The vector-based implementation offers a solution which is faster than the matrix based implementation, since memory allocation for matrices is not required. The space complexity for the matrix-based implementation is $O(m \cdot n)$ for calculating the local and global cost matrix, and the direction matrix. The vector-based computation principle is the same as for the matrix-based method, but instead of allocating matrices only two vectors are needed, and so the space complexity is reduced to $O(n)$. To obtain the DTW distance between the time series \mathbf{q} and \mathbf{c} the calculation of the j -th column of the global cost matrix $\mathbf{G}_{:,j}$ solely depends on the values of the previous column $\mathbf{G}_{:,j-1}$ and the respective distances of the time series \mathbf{c} and the j -th entry of \mathbf{q} . Since there is no dependence on the column $\mathbf{G}_{:,j-2}$, the algorithm overwrites $\mathbf{G}_{:,j-2}$ with the newly calculated vector $\mathbf{G}_{:,j}$. Figure 1 demonstrates this principle

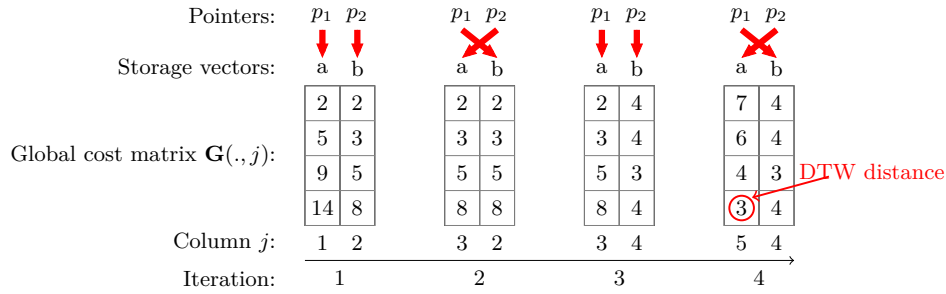


Figure 1: Iteratively overwriting vectors makes it obsolete to allocate matrices for DTW distance calculation.

with a simple example, and the following lines of code perform the DTW calculation for the same time series first via matrix based implementation (`dtw`) and second via vector based implementation (`dtw2vec`). The global cost matrix \mathbf{G} is also printed to compare it to the vectors illustrated in Figure 1.

```
R> Q <- c(3,4,5,6)
R> C <- c(1,3,3,5,6)
R> result <- IncDTW::dtw(Q,C)
R> result$gcm
```

```
      [,1] [,2] [,3] [,4] [,5]
[1,]    2    2    2    4    7
[2,]    5    3    3    4    6
[3,]    9    5    5    3    4
[4,]   14    8    8    4    3
```

```
R> dtw2vec(Q,C)$distance
```

```
[1] 3
```

In the first iteration step in Figure 1 the initial two vectors a and b are defined according to the DTW step pattern and are identical to the first two columns of \mathbf{G} . In the second iteration the pointers p_1 and p_2 switch the address, so that the new entries of $\mathbf{G}_{.,3}$ overwrite a (where p_2 points to) and b (where p_1 points to) stores the entries of $\mathbf{G}_{.,2}$ of the previous iteration. Finally after four iterations the DTW distance measure (red encircled) is given in the last row of the last vector, which is identical to the fifth column of \mathbf{G} . Algorithm 2 formalizes this principle for the general case.

Even though the information about the warping alignment is lost by applying the vector-based method, the warping path still can be constrained by the parameter `ws`, defining the Sakoe Chiba warping window size. To continue with the same time series we constrain the warping path to allow a maximum deviation of the time index of \mathbf{q} and \mathbf{c} of 1, so $|i - j| \leq 1$. Since the warping path needs to adapt slightly the calculated distance changes from 3 to 4.

Algorithm 2 Vector based implementation of DTW without allocating any matrices.

```

1: procedure VECTOR BASED DTW( $\mathbf{q} \in \mathbb{R}^{n \times O}$ ,  $\mathbf{c} \in \mathbb{R}^{m \times O}$ )
2:    $p1 \leftarrow \text{cumsum}(\text{dist}(q_1, \mathbf{c}))$  ▷ initial column of  $\mathbf{G}$ ,  $\mathbf{G}_{:,1}$ 
3:   for  $j$  in 2:m do
4:      $p2[1] \leftarrow \text{dist}(q_1, c_j) + p1[1]$ 
5:     for  $i$  in 2:n do
6:        $p2[i] \leftarrow \text{step}(\text{dist}(q_i, c_j), \min(p2[i-1], p1[i], p1[i-1]))$ 
7:     end for
8:      $ptmp \leftarrow p1$ 
9:      $p1 \leftarrow p2$ 
10:     $p2 \leftarrow ptmp$ 
11:  end for
12:  return  $p1[n]$ 
13: end procedure

```

```
R> IncDTW::dtw2vec(Q, C, ws = 1)$distance
```

```
[1] 4
```

“Early abandoning” is a pruning method to break calculations if the cheapest possible alignment of two time series hits an upper bound (set by the user). This method helps to lower the calculation runtime when comparing many time series. If the DTW algorithm hits this threshold the for-loop breaks and returns `NaN`. We continue the example and set the threshold to 2. Since no value in the fourth column of the global cost matrix is smaller or equal to 2, so $\mathbf{G}_{i,4} > 2 \forall i$, the calculation stops here and `NaN` is returned.

```
R> IncDTW::dtw2vec(Q, C, threshold = 2)$distance
```

```
[1] NaN
```

3.3. IncDTW for incremental DTW calculation

For the incremental calculation of DTW we can choose between (1) the matrix based implementation to get more information about the alignment of the two time series and to facilitate analyses of the warping paths and (2) the vector based implementation for a faster distance calculation. For the latter the initial column in Algorithm 2 is defined as the last column of the former calculated global cost matrix, the last pointer vector respectively. That is, instead of passing matrices as input to the incremental DTW function, only the last column vector of \mathbf{G} is passed for the vector based implementation. Further the class ‘`planedtw`’ and its methods deal as convenient wrapper functions around the vector based implementation. For a better understanding the following examples first walk through the more basic matrix based and vector based incremental update, and finally present the incremental update by hand of the ‘`planedtw`’ class.

We demonstrate the principle of incrementally updating the DTW global distance matrix and the distance measure by hand of the following example. We define the time series \mathbf{q} and \mathbf{c} , and calculate the initial alignment with `dtw()`.

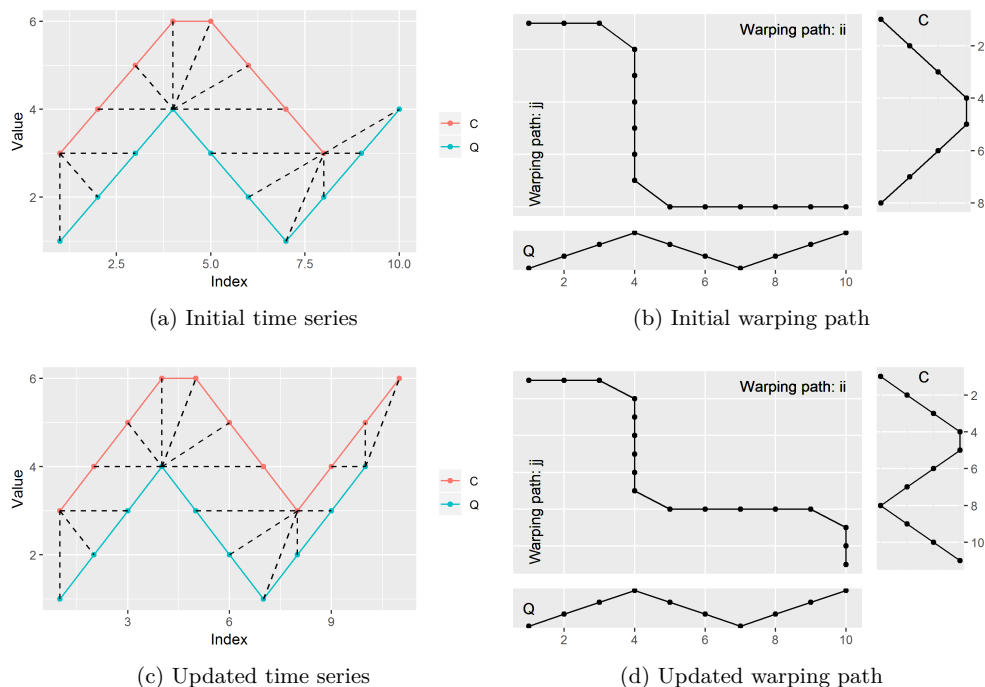


Figure 2: Initially \mathbf{c} and \mathbf{q} are aligned. The warping path has 4 vertical steps before the update of \mathbf{c} . The alignment is updated with an update of \mathbf{c} .

```
R> Q <- c(1:3, 4:1, 2:4)
R> C_initial <- c(1:3, 4, 4, 3:1) + 2
R> align_initial <- IncDTW::dtw(Q = Q, C = C_initial, return_wp = TRUE,
+   return_QC = TRUE, step_pattern = "symmetric1")
```

Figure 2a shows the time series and the aligned observations connected with dashed lines, and Figure 2b contains the same information but focuses on the warping path (the main plot). One can see that the last observation of \mathbf{c} is matched to the final six observations of \mathbf{q} . We plotted the results with `plot(align_initial, type = "warp")` and `type = "QC"` respectively.

With new observations of \mathbf{c} we can easily update the global cost matrix and the warping path by applying `idtw()` and compare the initial and updated versions of \mathbf{G} .

```
R> C_newObs <- Q[8:10] + 2
R> C_update <- c(C_initial, C_newObs)
R> align_inc <- IncDTW::idtw(Q = Q, C = C_initial, newObs = C_newObs,
+   gcm = align_initial$gcm, dm = align_initial$dm, return_wp = TRUE,
+   return_QC = TRUE, step_pattern = "symmetric1")
R> identical(align_inc$gcm[, 1:8], align_initial$gcm)
```

[1] TRUE

As expected the first eight columns of the updated \mathbf{G} and the initial \mathbf{G} are identical. Figure 2c and 2d show the updated alignment and warping path. Finally, we compare the DTW distance of the updated calculation with the one from scratch (again using the basic function `dtw()`) and see that they are equal:

```
R> align_scr <- IncDTW::dtw(Q = Q, C = C_update, return_wp = TRUE,
+   return_QC = TRUE, step_pattern = "symmetric1")
R> align_scr$distance - align_inc$distance
```

```
[1] 0
```

We continue with the former example and perform the incremental calculation with the vector based implementation with `idtw2vec()`. This function distinguishes between an initial calculation and the incremental by checking whether results of previous calculations are passed or not, particularly the argument `gcm_lc`.

```
R> alignV_init <- IncDTW::idtw2vec(Q = Q, newObs = C_initial, gcm_lc = NULL)
R> alignV_inc <- IncDTW::idtw2vec(Q = Q, newObs = C_newObs,
+   gcm_lc = alignV_init$gcm_lc_new)
```

Finally we compare the DTW distances of the incremental calculation (`idtw2vec()`) with the one from scratch (`dtw2vec()`) and their matrix based counterparts. As expected they are identical:

```
R> C_update <- c(C_initial, C_newObs)
R> alignV_scr <- IncDTW::dtw2vec(Q = Q, C = C_update)
R> c(align_scr$distance, align_inc$distance,
+   alignV_scr$distance, alignV_inc$distance)
```

```
[1] 16 16 16 16
```

Section 4.2 gives runtime comparisons for these update functions.

New observations for both time series

With the knowledge of the basics and main modules for incremental calculation of DTW, `idtw()` and `idtw2vec()`, we apply the functions `initialize_plane()` and `increment()` which are convenient wrappers around `idtw2vec()`. The former function performs the initial calculation of `idtw2vec()` and returns an object of class `'planedtw'`, whereas the latter function applies the incremental calculation of `idtw2vec()`. The package vignette (Leodolter 2021) discusses further methods for the S3 class `'planedtw'` that support the navigation in the plane of possible fits, which means to adjust incrementally the partial alignment of two time series.

If new observations for both time series are available, the update of the DTW calculation works in a consecutive fashion, similar to the case where only one time series is updated. The initial step is to apply `initialize_plane()` on the initial observations of \mathbf{c} and \mathbf{q} . Next we update the calculations for the first time series with `increment()`:

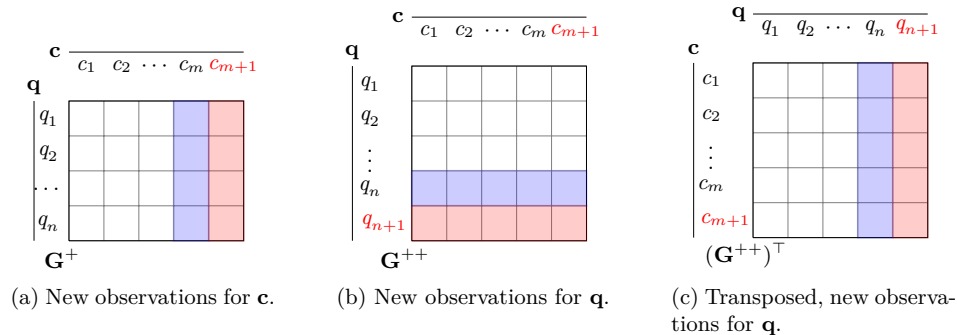


Figure 3: Incremental update of \mathbf{G} for new observations of \mathbf{c} and \mathbf{q} . The updated areas of \mathbf{G} are coloured in red and the areas storing the required input for the vector based update calculation are coloured blue.

```
R> x <- initialize_plane(Q = Q, C = C_initial)
R> print(x)
```

```
control:
  dist_method step_pattern nQ nC   ws reverse
    norm1     symmetric2 10  8 NULL  FALSE
```

```
DTW distance:
14
```

```
Normalized DTW distance:
0.7777778
```

```
R> x <- increment(x, newObs = C_newObs)
```

Figure 3a visualizes relevant sections of the updated global cost matrix \mathbf{G} . For a new observation of \mathbf{c} the new area of \mathbf{G} is colored red and the required column for the update in blue. Next we update \mathbf{G} for the new observations of \mathbf{q} . Again the red and blue rows in Figure 3b indicate the updated and required areas. So we switch places of \mathbf{q} and \mathbf{c} as input for `idtw2vec()` and proceed analogously. Also we need to switch the last column with the last row of the global cost matrix. Figure 3c illustrates that switching \mathbf{c} and \mathbf{q} and the `gcm_lr` with `gcm_lc` is the same as transposing \mathbf{G} . We could either switch the positions of these elements by hand and apply `idtw2vec()` directly, or apply the more convenient function `increment()` and set `direction = "Q"` to tell the function in which direction to update the last row and column of the global cost matrix:

```
R> Q_newObs <- rw(10)
R> x <- increment(x, newObs = Q_newObs, direction = "Q")
```

Finally we compare the results with the results from scratch and see that the calculated distance measures are equal:

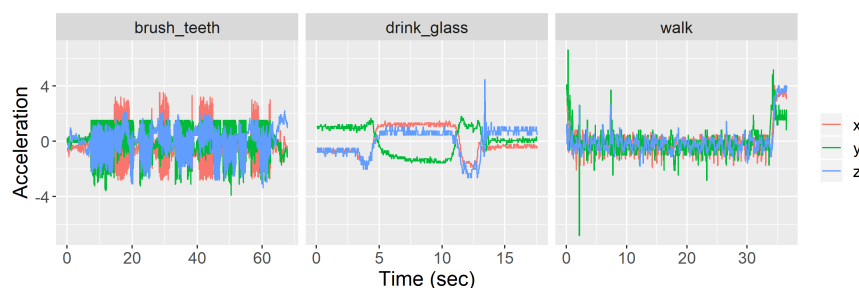


Figure 4: Typical accelerometer time series recorded while brushing teeth, drinking a glass, or walking.

```
R> x$distance - dtw2vec(c(Q, Q_newObs), c(C_initial, C_newObs))$distance
```

```
[1] 0
```

4. Applying IncDTW

This section demonstrates the applicability of **IncDTW** by (1) discussing a time series classification task for live data streams solved by either the traditional DTW implementation `dtw2vec()` or the incremental updating of DTW distances to speed up calculations with `idtw2vec()`, and (2) discussing runtime experiments that compare **IncDTW** with other R and Python packages.

In the following experiment we work with data sets (Bruno, Mastrogiovanni, Sgorbissa, Ver-nazza, and Zaccaria 2013) downloaded from UCI machine learning repository (Dheeru and Karra Taniskidou 2017). The data was collected by participants wearing a smart watch recording a 3-dimensional accelerometer signal with a sampling rate of 32 Hz. Among other actions the participants were asked to collect data during walking (`walk`), drinking a glass (`drink_glass`) and brushing teeth (`brush_teeth`), Figure 4 depicts examples of the three activities. The time series data of these experiments are included in the package **IncDTW**. The package documentation and vignette (Leodolter 2021) also include further experiments about time series clustering and scanning longer time series to detect similar representations of a shorter query pattern.

4.1. Incremental DTW update for live data

When applying data mining methods on live streams of data, it is mandatory that the computation time of the analysis is smaller than the time in between two consecutive observations. In this experiment we simulate the situation of dealing with data streams by iteratively including more observations of the time series into analysis. As soon as new observations are “recorded” we classify the time series streams by comparing their DTW distances to prototype patterns, so we need to update the DTW calculation for each set of new observations.

We start this experiment with determining representative centroid patterns for each of the recorded activities, stored in the accelerometer data sets `walk`, `drink_glass` and `brush_teeth`.

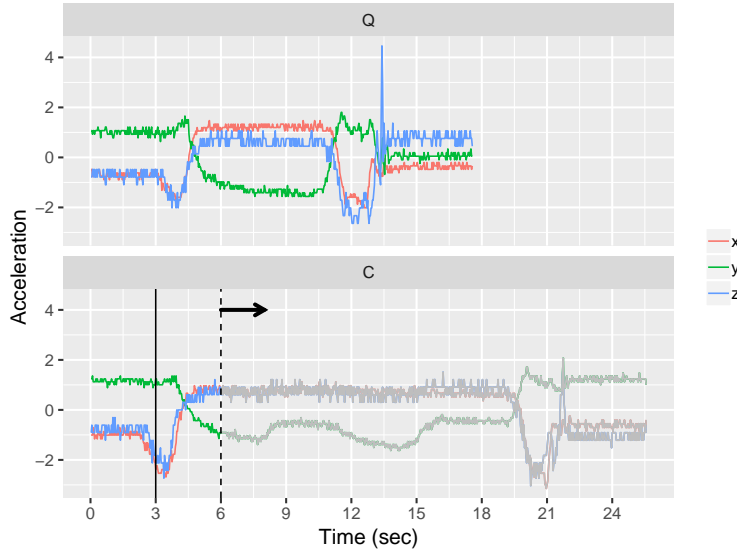


Figure 5: Iteratively increasing the observation window. As the dashed line moves to the right, more data is included in analysis and the DTW alignment is updated for the new observations.

We calculate these representatives with `IncDTW::dba()`, which is the DTW Barycenter Averaging method by Petitjean, Ketterlin, and Gançarski (2011) for averaging multiple time series that are non-linearly aligned by DTW.

Next we calculate the initial DTW distances for the first 100 observations (about 3 seconds) of each time series of the three data sets to the three centroids. Then we simulate the continuous recording of new observations and apply `idtw2vec()` to update the DTW distance measures, which requires to store the last columns of \mathbf{G} (see (2)) of the previous calculations. For comparing the computation times we fulfill the same classification task with `dtw2vec()`, and of course the classification results are identical. Figure 5 depicts this simulation of a data stream \mathbf{c} and the query time series \mathbf{q} , both selected from the `drink_glass` data set. This plot shows the situation after the initial step – the first three seconds are already observed (vertical solid line) – when \mathbf{c} has already been recorded for six seconds in total (the vertical dashed line). As the data stream continuously updates the dashed line moves to the right and more observations are included to the DTW alignment with \mathbf{q} .

Figure 6a plots the classification accuracy against the “observed” (used) percentage of the time series, and shows that the accuracy increases the more observations are recorded. Already about 75% are enough to reach an F1-score of 90%. We used 4-fold cross validation, where we calculated the representatives via `dba()` on one fold and classified the remaining 3 folds. Figure 6a shows aggregated results.

Figure 6b compares the computation times of `idtw2vec()` (incremental) and `dtw2vec()` (from scratch) to process one set of new observations, which we represent as the set of observations recorded within one second, so 3-dimensional time series with 32 rows (since originally recorded with 32Hz). The collection of these three data sets consists of 212 time series of different lengths. The calculation times depend on the length of the observation window and

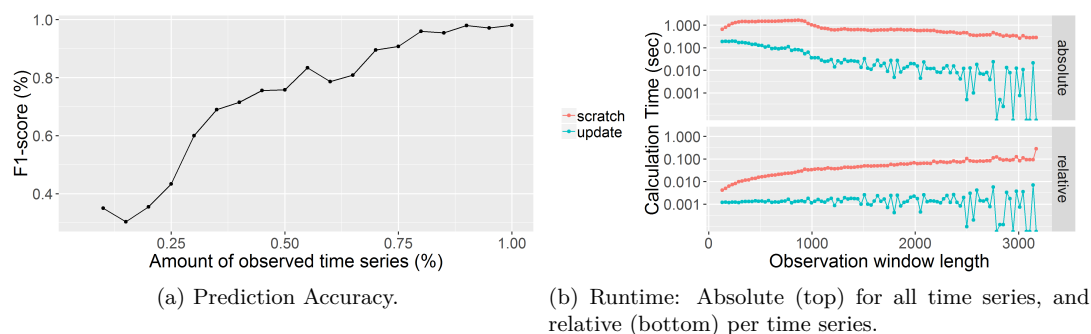


Figure 6: Prediction accuracy and computation time comparison for classifying multivariate time series of the data sets `walk`, `drink_glass` and `brush_teeth` by simulating to observe these time series live and update the prediction once per second.

the number of time series that are at least as long as the observation window. Since the time series are of different lengths, with increasing observation window, more and more time series can not be processed further until the observation window is equal to the length of the longest time series. For this reason the graph for “scratch” in Figure 6b (top) first rises and then drops continuously. All time series are at least 187 observations long and beyond this observation window length the shorter time series drop out of further analysis and so are not relevant for the total computation time. For clarification we also plot the relative times per time series in Figure 6b (bottom). It is worth mentioning that the y -axis are log-scaled.

We conclude that the incremental update can process about 7 to 108 times more time series than the calculation from scratch, dependent on the length of the time series, the observation window respectively. This exemplary data analysis task would not be solvable in time by applying `dtw2vec()` (which is vector-based implemented in C++ via **Rcpp**) since the calculation of DTW distances and classification takes longer than one second, which is the time in-between two sets of new observations. However, the incremental method with `idtw2vec()` is capable. As expected this experiment demonstrates the calculation time for the incremental step to be independent of the total length of the time series, see Figure 6b. We performed this experiment applying a single core of a 2.8 GHz and 16GB RAM laptop. If we split the work for this example across multiple cores `dtw2vec()` would manage the classification in time as well, however the relation of 7 to 108 remains the same, so the incremental solution is capable to deal with much more time series updates in less time.

4.2. Runtime comparisons

In the following we compare computation times for the 3 data analysis tasks: (1) the incremental update for new observations, (2) single DTW computation for two time series, and (3) computing the matrix of pairwise DTW distances for a set of time series. Further, we also compare **IncDTW** with Python packages for the second task, since this is probably the most generic and most applied use case. To compare the calculation times of R packages we use the package **microbenchmark** (Mersmann 2019). For comparisons to Python packages we measure the wall clock time. To the best of our knowledge we set the parameters of all functions so that a fair computation time comparisons is guaranteed. So we omit to

return additional output objects (like the warping path) which obviously would cause higher computation times. All runtime experiments were performed on a standard laptop computer with 2.8 GHz and 16GB RAM. We applied the following versions of the respective packages (please see Section 1 and Table 1 for more details about the packages):

- R: **IncDTW** (1.0.4), **dtw** (1.22-3), **dtwclust** (5.5.6), **rucrdtw** (0.1.4), **parallelDist** (0.2.4)
- Python: **cydtw** (0.1.4), **dtaidistance** (1.2.3), **dtw** (1.4.0).

Incremental update of DTW

This paper emphasizes methods for accelerating DTW calculations and demonstrates how to apply the incremental DTW calculation for updating existing results for new observations (Section 2.1 and 4.1). The following experiment underpins that this principle of recycling former calculated results is a considerable faster approach to compute the DTW distance measure. For this experiment we simulate the situation of continuously recording new observations and compare the runtime for the incremental calculation with a traditional calculation from scratch. Figure 7a shows the results. Each red point is the median of 100 computations of the DTW distance with `dtw2vec()` of two univariate time series, both of the respective length given at the x -axis. The blue points visualize the median computation time for one incremental step (via `idtw2vec()`), so one new observation of \mathbf{c} , and \mathbf{q} of length as given by the x -axis. Both axes are in log scale.

Single computations

Figure 7b depicts the runtime comparison in a log-scale. The only two methods using a vector-based implementations (as discussed in Section 3.2) are `rucrdtw::ucrdtw_vv()` and `IncDTW::dtw2vec()`, and these are considerably faster than the remaining functions. To guarantee a fair comparison we set the step pattern to “symmetric1” (since `rucrdtw::ucrdtw_vv()` only supports “symmetric1”) and the warping window size equal 10 for all functions.

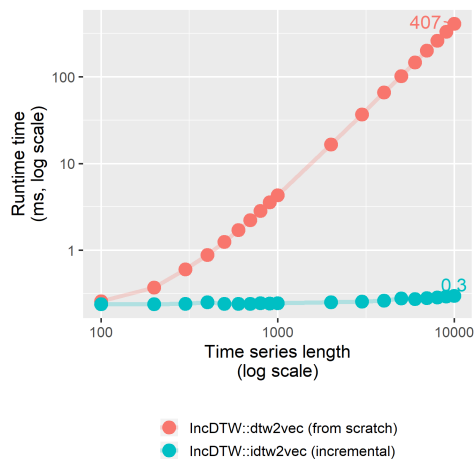
Compute a distance matrix

Time series clustering is a typical task in time series analysis and data mining. Time series clustering based on the DTW distance measure requires a distance matrix of pairwise DTW distances. The function `IncDTW::dtw_dismat()` helps to get this matrix for a list of univariate or multivariate time series of possibly different lengths. The calculations can be performed single threaded (`ncores = 1`) or multithreaded.

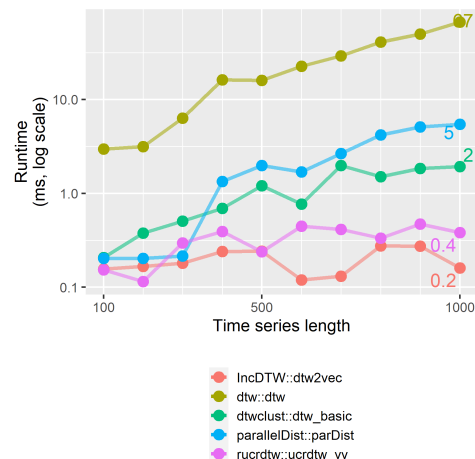
We compare the runtimes for calculating distance matrices for a set of 500 time series of varying lengths and also set the window size parameter to 10. Figure 7c depicts the runtimes, where `dtw_dismat_1()` is the standard function `dis_mat()` without parallelization. `dtw_dismat_3Rcpp()` splits the work via `RcppParallel` and `dtw_dismat_3R()` uses the package `parallel`, both with three cores (`ncores = 3`). For short time series `parDist_3()` is up to 10 times faster than `dtw_dismat_3Rcpp()` and for long time series it’s the other way round (about 17 times faster).

Comparison with Python

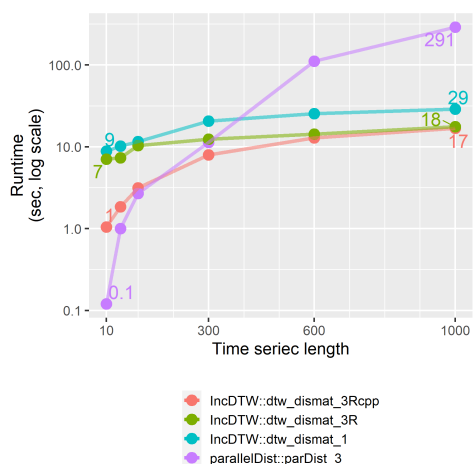
For many data analysis tasks R and Python are interchangeable and it is just a matter of



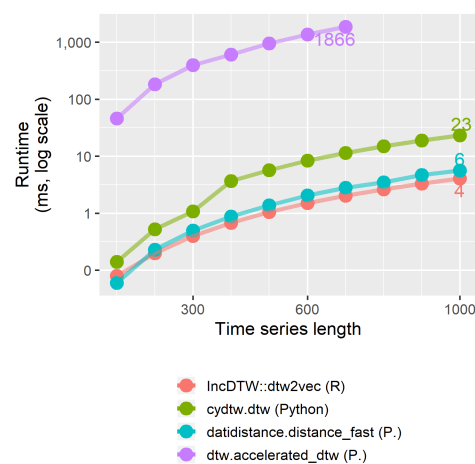
(a) Incremental vs. from scratch.



(b) Single DTW computations for two time series.



(c) Matrices of pairwise DTW distances for a list of time series.



(d) Single DTW without warping window compared with Python packages.

Figure 7: Runtime comparisons for different data analysis tasks.

taste which to prefer. So, we compare the runtimes for calculating the DTW distance across these two platforms. For each of the time series lengths we measured the wall clock time for 100 DTW computations in the respective programming language environment², and averaged it. To guarantee a fair comparison we omit the warping window parameter since the function `cydtw.dtw()`³ does not support warping windows. Figure 7d shows the results in log scale. The functions `datidistance.distance_fast()` and `cydtw.dtw()` both are functions

²We also performed the experiment by calling the Python functions inside of R via `reticulate` (Ushey, Allaire, and Tang 2021), which caused a computation overhead.

³We notate R and Python functions according to their syntax: `package::function()` in R and `package.function()` in Python.

written in C, via **Cython**, but only the former is vector based and so it is comparable fast as `IncDTW::dtw2vec()`.

5. Conclusion

This paper discusses the incremental calculation of the widely applied DTW distance measure (Fu 2011). We present the R package **IncDTW** (Leodolter 2021) – current version 1.1.4.3 available from the Comprehensive R Archive Network at <https://CRAN.R-project.org/package=IncDTW> – that mainly focuses on fast R functions for vector based and incremental DTW computation. **IncDTW** also offers functions for familiar time series analysis tasks, as time series clustering and pattern recognition. Section 4.1 showcases how to apply **IncDTW** to classify three dimensional time series in a simulated live stream setting, and why the incremental calculation of DTW is capable to process 7 to 108 times more data.

Due to the intensive computational costs of DTW, we put a special emphasis on accelerating our algorithms. Consequently, **IncDTW** transfers the most intensive computations to C++ via **Rcpp** and stresses on the one hand the vector based implementation, and on the other hand the principle of the incremental calculation of DTW, by recycling previous calculation results. Section 4.2 demonstrates the benefits of these acceleration methods using runtime comparisons for various settings. Further accelerating methods as lower bounding (Keogh, Wei, Xi, Lee, and Vlachos 2006; Rath and Manmatha 2003a) and early abandoning methods are also applied and discussed in more detail in the package vignette (Leodolter 2021).

Apart from stream processing, computation time is also key whenever relatively short query patterns must be detected in longer time series, which usually requires a large number of comparisons between many segments of the longer time series and the query pattern. For example, the Caterpillar algorithm presented byLeodolter, Brändle, and Plant (2018) scans long time series to detect patterns which are possibly warped or of different lengths than a query pattern, based on a combination of incremental DTW calculation and the Minimum Description Length. The incremental calculation of DTW enables the Caterpillar algorithm to search the space of possible fits runtime efficiently. So, the R package **IncDTW** and its functions can serve as components for building pattern recognition algorithms.

Future developments for **IncDTW** will incorporate a parallelized implementation of `dba()` and a user-friendly solution for applying lower bounding, which is currently only implemented as part of `rundtw()`.

References

- Allaire JJ, François R, Ushey K, Vandenbrouck G, Geelnard M, Intel (2021). **RcppParallel**: *Parallel Programming Tools for Rcpp*. R package version 5.1.4, URL <https://CRAN.R-project.org/package=RcppParallel>.
- Behnel S, Bradshaw R, Citro C, Dalcin L, Seljebotn DS, Smith K (2011). “Cython: The Best of Both Worlds.” *Computing in Science Engineering*, **13**(2), 31–39. doi:10.1109/mcse.2010.118.
- Berndt DJ, Clifford J (1994). “Using Dynamic Time Warping to Find Patterns in Time Se-

- ries.” In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS’94, pp. 359–370. AAAI Press. URL <http://dl.acm.org/citation.cfm?id=3000850.3000887>.
- Boersch-Supan P (2016). “**rucrdtw**: Fast Time Series Subsequence Search in R.” *The Journal of Open Source Software*, **1**, 1–2. doi:10.21105/joss.00100.
- Bruno B, Mastrogiovanni F, Sgorbissa A, Vernazza T, Zaccaria R (2013). “Analysis of Human Behavior Recognition Algorithms Based on Acceleration Data.” In *IEEE International Conference on Robotics and Automation 2013*, pp. 1602–1607. IEEE.
- Dheeru D, Karra Taniskidou E (2017). “UCI Machine Learning Repository.” URL <http://archive.ics.uci.edu/ml>.
- Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh E (2008). “Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures.” *Proceedings of the VLDB Endowment*, **1**(2), 1542–1552. doi:10.14778/1454159.1454226.
- Dixon S (2005). “An On-Line Time Warping Algorithm for Tracking Musical Performances.” In *IJCAI*, pp. 1727–1728.
- Eckert A (2018). **parallelDist**: *Parallel Distance Matrix Computation Using Multiple Threads*. R package version 0.2.4, URL <https://CRAN.R-project.org/package=parallelDist>.
- Eddelbuettel D, François R (2011). “**Rcpp**: Seamless R and C++ Integration.” *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.
- Eddelbuettel D, Sanderson C (2014). “**RcppArmadillo**: Accelerating R with High-Performance C++ Linear Algebra.” *Computational Statistics & Data Analysis*, **71**, 1054–1063. doi:10.1016/j.csda.2013.02.005.
- Fu T (2011). “A Review on Time Series Data Mining.” *Engineering Applications of Artificial Intelligence*, **24**(1), 164–181. doi:10.1016/j.engappai.2010.09.007.
- Giorgino T, *et al.* (2009). “Computing and Visualizing Dynamic Time Warping Alignments in R: The **dtw** Package.” *Journal of Statistical Software*, **31**(7), 1–24. doi:10.18637/jss.v031.i07.
- Keogh E (2002). “Exact Indexing of Dynamic Time Warping.” In *VLDB’02: Proceedings of the 28th International Conference on Very Large Databases*, pp. 406–417. Elsevier.
- Keogh E, Wei L, Xi X, Lee SH, Vlachos M (2006). “LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures.” In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pp. 882–893. VLDB Endowment.
- Keogh EJ, Pazzani MJ (2000). “Scaling up Dynamic Time Warping for Datamining Applications.” In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’00, pp. 285–289. ACM, New York. doi:10.1145/347090.347153.

- Kwankhoom W, Muneesawang P (2017). “An Incremental Dynamic Time Warping for Person Re-Identification.” In *14th International Joint Conference on Computer Science and Software Engineering 2017*, pp. 1–5. IEEE.
- Leodolter M (2021). *IncDTW: Incremental Calculation of Dynamic Time Warping*. R package version 1.1.4.3, URL <https://CRAN.R-project.org/package=IncDTW>.
- Leodolter M, Brändle N, Plant C (2018). “Automatic Detection of Warped Patterns in Time Series: The Caterpillar Algorithm.” In *IEEE International Conference on Big Knowledge 2018*, pp. 423–431. doi:10.1109/icbk.2018.00063.
- Maus V, Câmara G, Appel M, Pebesma E (2019). “*dtwSat*: Time-Weighted Dynamic Time Warping for Satellite Image Time Series Analysis in R.” *Journal of Statistical Software*, **88**(5), 1–31. doi:10.18637/jss.v088.i05.
- Meert W (2017). *dtaidistance*. URL <https://pypi.org/project/dtaidistance/>.
- Mersmann O (2019). *microbenchmark: Accurate Timing Functions*. R package version 1.4-7, URL <https://CRAN.R-project.org/package=microbenchmark>.
- Mori A, Uchida S, Kurazume R, Taniguchi R, Hasegawa T, Sakoe H (2006). “Early Recognition and Prediction of Gestures.” In *18th International Conference on Pattern Recognition (ICPR’06)*, volume 3, pp. 560–563. doi:10.1109/icpr.2006.467.
- Oregi I, Pérez A, Del Ser J, Lozano JA (2017). “On-Line Dynamic Time Warping for Streaming Time Series.” In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 591–605. Springer-Verlag.
- Petitjean F, Ketterlin A, Gançarski P (2011). “A Global Averaging Method for Dynamic Time Warping, with Applications to Clustering.” *Pattern Recognition*, **44**(3), 678–693. doi:10.1016/j.patcog.2010.09.013.
- Rabiner L, Juang BH (1993). *Fundamentals of Speech Recognition*. Prentice-Hall, Upper Saddle River.
- Rabiner L, Rosenberg A, Levinson S (1978). “Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition.” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **26**(6), 575–582. doi:10.1109/tassp.1978.1163164.
- Rakthanmanon T, Campana B, Mueen A, Batista G, Westover B, Zhu Q, Zakaria J, Keogh E (2012). “Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping.” In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 262–270. ACM. doi:10.1145/2339530.2339576.
- Rath TM, Manmatha R (2003a). “Lower-Bounding of Dynamic Time Warping Distances for Multivariate Time Series.” *MM 40*, University of Massachusetts Amherst.
- Rath TM, Manmatha R (2003b). “Word Image Matching Using Dynamic Time Warping.” In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2003*, volume 2, pp. II–II. IEEE. doi:10.1109/cvpr.2003.1211511.

- R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- Rouanet P (2014). *dtw*. URL <https://pypi.org/project/dtw/>.
- Sakoe H, Chiba S (1978). “Dynamic Programming Algorithm Optimization for Spoken Word Recognition.” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **26**(1), 43–49. doi:10.1109/tassp.1978.1163055.
- Sakurai Y, Faloutsos C, Yamamuro M (2007). “Stream Monitoring under the Time Warping Distance.” In *IEEE 23rd International Conference on Data Engineering 2007*, pp. 1046–1055. IEEE.
- Sarda-Espinosa A (2019). *dtwclust: Time Series Clustering Along with Optimizations for the Dynamic Time Warping Distance*. R package version 5.5.6, URL <https://CRAN.R-project.org/package=dtwclust>.
- Tavenard R (2017). *cydtw*. URL <https://pypi.org/project/cydtw/>.
- Tormene P, Giorgino T, Quaglini S, Stefanelli M (2008). “Matching Incomplete Time Series with Dynamic Time Warping: An Algorithm and an Application to Post-Stroke Rehabilitation.” *Artificial Intelligence in Medicine*, **45**(1), 11–34. doi:10.1016/j.artmed.2008.11.007.
- Ushey K, Allaire JJ, Tang Y (2021). *reticulate: Interface to Python*. R package version 1.20, URL <https://CRAN.R-project.org/package=reticulate>.
- Van Rossum G, et al. (2011). *Python Programming Language*. URL <https://www.python.org/>.
- Wang X (2011). “A Fast Exact k -Nearest Neighbors Algorithm for High Dimensional Search Using k -Means Clustering and Triangle Inequality.” In *The 2011 International Joint Conference on Neural Networks*, pp. 1293–1299. IEEE.

Affiliation:

Maximilian Leodolter
Austrian Institute of Technology
Center for Mobility Systems
1210 Wien, Austria
E-mail: maximilian.leodolter@gmail.com

Paper D

Rotation Invariant GPS Trajectory Mining

License Information

This paper is licensed under the Creative Commons Attribution 4.0 International License¹. The original work was not changed for presentation in this thesis.

¹<https://creativecommons.org/licenses/by/4.0/>



Rotation invariant GPS trajectory mining

Maximilian Leodolter¹  · Claudia Plant^{2,3} · Norbert Brändle¹

Received: 27 December 2021 / Revised: 1 February 2023 / Accepted: 6 March 2023
© The Author(s) 2023

Abstract

Mining of GPS trajectories of moving vehicles and devices can provide valuable insights into urban systems, planning and operational applications. Understanding object motion often requires that the spatial-temporal matching of trajectories be invariant to shifting, scaling and rotation. To this end, Procrustes analysis enables to transform one data set of a trajectory to represent another set of data as closely as possible. We propose a novel shift-scale-rotation invariant Procrustes distance metric based on the Kabsch algorithm, which calculates the optimal rotation matrix by minimizing the root-mean squared deviation between two paired sets of points of trajectories or trajectory segments. We present two novel runtime efficient algorithms which are based on our proposed distance metric: 1) the sliding-shifting-scaling-Kabsch-rotation (S3KR) algorithm for detecting recurring short query patterns in longer motion trajectories and 2) a novel time series subsequence clustering algorithm to group GPS trajectory data and to discover prototypical patterns. We demonstrate the potential of our proposed sliding Procrustes analysis algorithms by applying it on real-world GPS trajectories collected in urban and rural areas from different transport modes, as well as on nautical GPS trajectories. We also demonstrate that our methods outperform the state of the art in accuracy and runtime on synthetic and real world data.

Keywords Trajectory · GPS · Clustering · Distance measure

1 Introduction

The increasingly wide use of mobile devices and sensors leads to a flood of machine-generated trajectory data about moving people, vehicles, vessels and other objects. A plethora of work has been developed in different domains taking advantage of this new wealth of motion data, proposing techniques for analyzing motion trajectory data, e.g. [25]. Understanding motion

Maximilian Leodolter
maximilian.leodolter@gmail.com

- ¹ Austrian Institute of Technology, Vienna, Austria
- ² Faculty of Computer Science, University of Vienna, Vienna, Austria
- ³ ds:Univie, University of Vienna, Vienna, Austria

often requires that the spatial-temporal matching of trajectories be invariant to shifting, scaling and rotation.

Traditional time series distances lack the capability to discover rotation invariant similarities, and therefore require some form of preprocessing of the time series [3, 19, 24]. As such preprocessing methods are sensitive to noise (as we demonstrate in Section 4.1), we propose a novel distance method based on Procrustes analysis [5] and demonstrate that it outperforms traditional methods from literature.

Procrustes analysis can be used to transform one set of data (e.g. a trajectory or shape) to represent another set of data as closely as possible. The Kabsch algorithm [6] solves the mathematical problem of finding the rotation matrix which minimizes the distance of the rotated shape to another shape. In order to perform shape comparisons which are invariant to translation/shift, scale and rotation for each time point of a long trajectory, a runtime efficient sliding algorithm is essential.

A disadvantage of traditional Procrustes analysis is that it is computationally too expensive for exhaustive time series analysis. We tackle this complexity problem with a new distance measure – derived from Procrustes analysis – which is suitable for integration in a runtime efficient algorithm. We also demonstrate that our new distance measure finds similarity structures in a time series database identical to traditional Procrustes analysis.

We propose a novel efficient sliding technique for Procrustes analysis based on the Kabsch algorithm, denoted as the S3KR algorithm (sliding-shifting-scaling-Kabsch-rotation). S3KR compares a query trajectory q with all sub-trajectories x of the long trajectory, as illustrated in Fig. 1: The left part of Fig. 1 shows the color-coded GPS trajectory of a car driving eastwards, where the colors along the trajectory indicate the distance to the short query trajectory q (a sharp left curve). In order to calculate the distance value for a point of the trajectory, we extract the sub-trajectory x starting at the point where the length of x equals the length of q . We shift, scale and rotate x to achieve y , and measure the distance of y to q . Yellow colors indicate low distance to the query trajectory, i.e. a sharp left turn. Conversely, black indicates large distance to the query trajectory, such as the right turn at the beginning of the trajectory.

The S3KR algorithm relies on a novel rotation invariant distance measure which is also a metric. Having the appealing features of a metric, notably the triangle inequality, means that this distance measure can be used in data structures enabling efficient k-NN or range queries, such as, for example, M-Trees [2]. We also introduce an algorithm for clustering segments of trajectories to discover typical structures and prototypical patterns in GPS trajectories. This algorithm relies on our proposed distance metric and the sliding algorithm S3KR.

We summarize our contributions: (1) a new rotation-scaling-shifting invariant Procrustes distance metric (see Section 3), (2) a sliding algorithm for efficient scanning of long time series (see Sections 3.2 and 3.3), (3) an algorithm based on spectral clustering for time series



Fig. 1 GPS trajectory of a car driving eastwards, color coded according to the shift-scale-rotation-invariant distance to the query pattern q which represents a sharp left turn

subsequence clustering (see Section 3.4), (4) experiments on synthetic random walks and trajectories representing handwritten characters (see Section 4), as well as two case studies on real-world urban and maritime GPS trajectories (see Section 5). We compare our methods with traditional time series distances and related methods from literature developed for similar problems. Our experiments demonstrate that our novel distance measure outperforms traditional time series distances on trajectory mining problems.

2 Procrustes analysis

In order to make this paper self-contained and to introduce our notation (Table 1) we briefly recapitulate the essentials of Procrustes analysis. We denote the mean-shift operator $\tau_\mu : \mathbf{R}^{n \times m} \rightarrow \mathbf{R}^{n \times m}$ as

$$\tau_\mu(x_{ij}) := x_{ij} - \mu_j, \tag{1}$$

where τ_μ is the typical shift operator in Procrustes analysis such that the shifted x has zero mean in all dimensions. For a multivariate time series x we notate the scaling operator $\gamma : \mathbf{R}^{n \times m} \rightarrow \mathbf{R}^{n \times m}$ as

$$\gamma(x_{ij}) := x_{ij} \cdot \left(\sum_j \sigma_j^2\right)^{-1/2} = x_{ij} \cdot s. \tag{2}$$

The scaling factor s is equal to the inverse of the root mean squared deviation of a trajectory to its mean, thus providing information about the extension of x in all dimensions.

Given two $n \times m$ -dimensional time series x and y , Kabsch' algorithm [6] solves the constrained orthogonal Procrustes problem [5] to find the rotation matrix R minimizing the Euclidean norm $\|x - y \cdot R\|$ by:

$$\begin{aligned} H &= x^T \cdot y && \in \mathbf{R}^{m \times m} \\ H &= U \cdot D \cdot V && \text{SVD (Singular Value Decomposition)} \\ d &= \det(V \cdot U^T) && \text{determinant} \\ E &= \text{diag}(1, \dots, 1, \text{sign}(d)) && \text{diagonal matrix of } m - 1 \text{ ones and } \text{sign}(d) \\ R &= V \cdot E \cdot U^T \end{aligned} \tag{3}$$

Since we want to preserve the direction of a trajectory, we concentrate on rotation matrices only, and therefore do not allow R to be a reflection matrix. Otherwise, a GPS trajectory representing a left turn could be reflected to perfectly fit a right turn.

For x and y of equal lengths we define the rotation operator $\pi : \mathbf{R}^{n \times m} \times \mathbf{R}^{n \times m} \rightarrow \mathbf{R}^{n \times m}$:

$$\pi(x, y) := y \cdot R, \tag{4}$$

where R is the rotation matrix according to (3).

Table 1 Notation

x, y, q, f multivariate time series $\in \mathbf{R}^{n \times m}$	R rotation matrix
x_{ij}, x_i the j -th or all dimensions of x at time i	x^T transposed matrix
τ, γ, π shift, scaling and rotation operator	\cdot matrix/scalar multiplication
μ, σ, s mean, standard deviation, scaling factor	H matrix, R depends on H
$\{y\}_{i,n}$ segment of y of length n , starts at time i	$\ \cdot\ $ Euclidean norm

The distance function δ which is invariant to scaling, shifting and rotation is defined as $\delta : \mathbf{R}^{n \times m} \times \mathbf{R}^{n \times m} \rightarrow \mathbf{R}_+$:

$$\delta(x, y) := \|\tau_\mu(\gamma(x)) - \pi(\tau_\mu(\gamma(x)), \tau_\mu(\gamma(y)))\|. \quad (5)$$

3 Sliding procrustes analysis

Performing distance calculations between many time series segments is runtime expensive. In the following we propose slight adaptations to the operators from Section 2 which enable the sliding algorithm to reduce the number of required operations and thus save calculation time.

3.1 Operators for sliding distance

We accompany the introduction of the computation operators with an illustration of two GPS trajectories in Fig. 2a and b, which are two segments of the original trajectory presented in Fig. 1. Before we calculate the distance between these two trajectories x and y , we shift both of them to the origin. We define the zero-shift operator (that shifts a time series such that it starts in the origin¹) $\tau_0 : \mathbf{R}^{n \times m} \rightarrow \mathbf{R}^{n \times m}$ for a multivariate time series x as follows:

$$\tau_0(x_{ij}) := x_{ij} - x_{0j}. \quad (6)$$

Figure 2c) depicts the shifted trajectory segments $\tau_0(x)$ and $\tau_0(y)$. Both shifted segments start at the origin. Next we apply γ (2) to scale the trajectories – Fig. 2d) shows $\gamma(\tau_0(\cdot))$ for x and y . The shapes remain almost identical to those from Fig. 2c), but the extensions in both dimensions have changed, and also the scales of the axes have changed. We remark that due to the fact that σ is invariant to translation, γ and τ_0 are commutative. It therefore makes no difference whether γ is applied first and followed by τ_0 , or vice versa. Before measuring the distance between x and y , we unify their orientation in a final step. We rotate both segments to minimize their abbreviation to one and the same reference time series (in this example we set $f_i = (0, 1) \forall i$, cf. (7)). Figure 2e) visualizes the rotated trajectory segments (after scaling and shifting) and finally shows the similarity of x and y , meaning that both segments are left turns. To achieve the results from Fig. 2e) we applied a rotation function defined as follows: for a time series y and a fixed reference time series f , we define the rotation function $\pi^f : \mathbf{R}^{n \times m} \rightarrow \mathbf{R}^{n \times m}$ which rotates y to minimize $\|f - y \cdot R\|$:

$$\pi^f(y) := y \cdot R, \quad (7)$$

where R is the rotation matrix according to (3) for f instead of x . Consequently, for a comparison of two time series x and y , π^f rotates x and y independently of each other, but dependent on the reference f . Section 3.3 discusses the choice of f and how this contributes to reducing the runtime complexity of the rotation from linear to constant.

Next we calculate the distance by summing up the squared lengths of the dashed lines in Fig. 2f), which is the Euclidean distance of the shifted, scaled and rotated time series. Formally, we define the shift-scaling-rotation invariant distance metric $\delta^f : \mathbf{R}^{n \times m} \times \mathbf{R}^{n \times m} \rightarrow \mathbf{R}_+$:

$$\delta^f(x, y) := \|\pi^f(\tau_0(\gamma(x))), \pi^f(\tau_0(\gamma(y)))\|. \quad (8)$$

¹ The benefit of τ_0 will become clear in Section 4, where we compare the performance of the zero-shift and the mean-shift operator

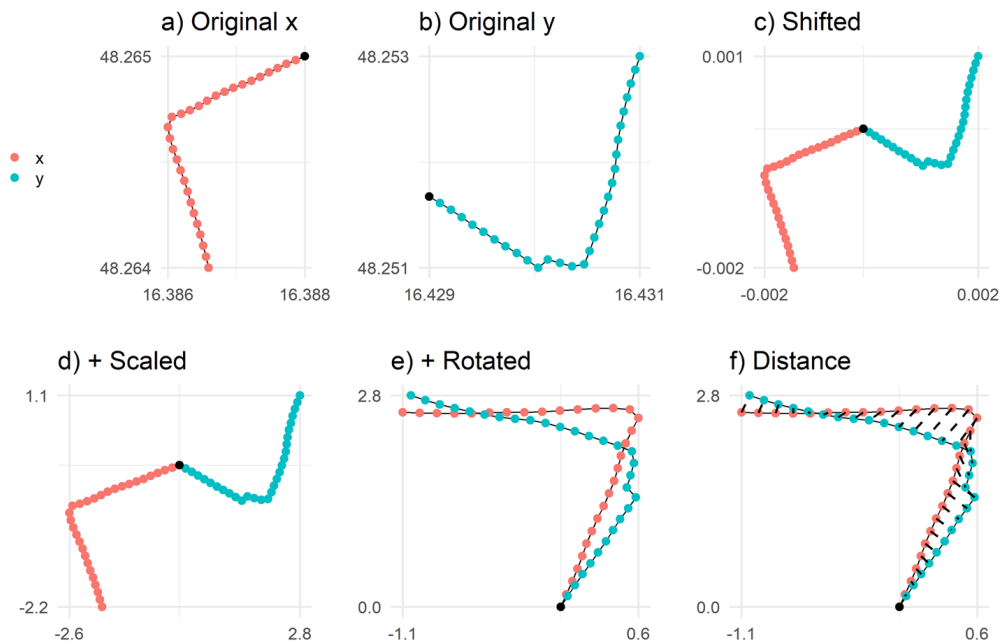


Fig. 2 a) and b) are two exemplary trajectory segments extracted from the path in Fig. 1 with original longitude and latitude scaling and orientation. c) The segments shifted to the origin (0, 0). d) The shifted and scaled segments. e) After shifting and scaling, the trajectory segments are rotated. Regarding direction: all the segments start at the black colored points

Appendix B proves that δ^f is a metric. Hence, δ^f is symmetric, and $\delta^f(x, y) = 0 \Leftrightarrow x$ and y are equal after scaling, shifting and rotating to fit f . Further δ^f fulfills the triangle inequality. Section 4.3 demonstrates how this is beneficial for standard data mining tasks as a nearest neighbor search.

3.2 Sliding distance: s3kr

The sliding distance s3kr enables to efficiently perform sliding pattern recognition (e.g. k-NN and ϵ -queries), and to efficiently compare time series of different lengths. To compare a shorter query q with segments of a longer time series y , s3kr executes this in a sliding fashion and calculates δ^f for every point of time. If $q \in \mathbf{R}^{n_q \times m}$ and $y \in \mathbf{R}^{n_y \times m}$ are two time series of different lengths $n_q < n_y$, we measure the distance between segments of y , $\{y\}_{i,n_q}$, and q by sliding a window of length n_q along y . The result is a vector of distances. We define the sliding-shifting-scaling-Kabsch-rotation function s3kr: $\mathbf{R}^{n_q \times m} \times \mathbf{R}^{n_y \times m} \rightarrow \mathbf{R}_+^{n_y - n_q + 1}$:

$$\text{s3kr}(q, y)_i := \delta^f(q, \{y\}_{i,n_q}). \tag{9}$$

Compared to a traditional distance computation, s3kr does not return a single distance value, but a vector of distances. Each entry of the distance vector represents the distance δ^f of q to a segment of y . Figure 1 shows a color coded GPS trajectory, where a GPS position is colored according to the distance between q (top right in Fig. 1) and the segment starting at this position. The vector of distances (bottom right in Fig. 1) is the result of s3kr.

To find the section of the longer time series that best fits the shorter time series q , we are only interested in the minimum of the vector of distances. For this purpose we define the

minimum distance as $\min_s3kr: \mathbf{R}^{n_q \times m} \times \mathbf{R}^{n_y \times m} \rightarrow \mathbf{R}_+$

$$\min_s3kr(q, y) := \min(s3kr(q, y)). \tag{10}$$

When one is only interested in finding the segment with minimum distance (1-NN, nearest neighbor search), it is easily possible to identify multiple unnecessary computation steps. To elaborate this, we first present an example to discuss early abandoning with a fixed threshold ϵ , and then show the adjustments for a 1-NN search. Consider the example of two univariate time series $x = (9, 1, 2, 4, 5, 9, 1, 2, 10)$ and $q = (1, 2, 3)$ and an ϵ -query looking for segments of x with a distance to q smaller than $\epsilon = 3$. The first segment is $\{x\}_{1,3} = (9, 1, 2)$ and has a Euclidean distance to q of $\sqrt{8^2 + 1^2 + 1^2} = \sqrt{66} > \epsilon$. This segment has a distance larger than the threshold $\epsilon = 3$ and is therefore not relevant. The question is whether it is possible to recognize this at an early stage before having finished the entire distance calculation. Here we want to stress the monotonicity of the Euclidean distance for time series: The Euclidean distance of two segments of two univariate time series x and y is always smaller or equal to the distance of the same segments extended by any number of observations ($k \geq 0$):

$$\begin{aligned} E(\{x\}_{i,n}, \{y\}_{j,n}) &= \sqrt{\sum_{m=1}^n (x_{i+m} - y_{j+m})^2} \leq \\ &\leq \sqrt{\sum_{m=1}^{n+k} (x_{i+m} - y_{j+m})^2} = d_E(\{x\}_{i,n+k}, \{y\}_{j,n+k}) \quad \forall i, j, n, k. \end{aligned} \tag{11}$$

The same applies to multivariate time series, and to the squared Euclidean distance. It is therefore also possible to compare the squared Euclidean distance with ϵ^2 , and one can observe that already the first element in the sum of the distance calculation is above the threshold: $8^2 > 3^2 = \epsilon^2$. Hence due to (11) one can already abandon the calculation for index $i = 1$ after having only compared the first elements of $\{x\}_{1,3}$ and q . In fact, for this example we only need to finish the distance calculation for $\{x\}_{i,3}$ where $i = 2, 3$ and 7 .

For the 1-NN search the threshold ϵ stores the distance to the nearest neighbor found so far. And ϵ is updated as soon as a segment is found which has a smaller distance than all segments evaluated so far – in this example after finishing the distance calculation for $\{x\}_{2,3}$, $\epsilon^2 = 1$. Consequently, in this example, we can early abandon the distance calculations for all remaining segments after hitting the threshold $\epsilon = 1$. For example, for $x_{3,3}$ the distance calculation already hits $\epsilon = 1$ after the first element. The same applies for $x_{i,3}$ for all $i > 2$ except for $i = 7$. $x_{7,3} = (1, 2, 10)$, so the first two elements are identical to q , and therefore the distance calculation must complete to recognize that the distance exceeds ϵ .

Early abandoning can help to drastically reduce the number of computation operations. Since δ^f is a metric, and applies d_E after shifting, scaling and rotation, (11) also applies for δ^f . Section 3.3 and Algorithm 2 give details about our implementation of early abandoning for δ^f .

3.3 Sliding algorithm: S3KR

Comparison of the GPS trajectories x and the query time series q in Fig. 1 rises the following questions: Which segments in x are similar to q , and which segments in x are dissimilar to q ? And, how can we quantify these similarities? Answering both questions requires to calculate distances of the separate segments of x and q . To get a similarity measure independent of scale, shift and rotation, we apply δ^f . Hence, calculating $s3kr(q, x)$ answers the questions.

The naive brute force approach to calculate $s3kr(q, x)$ is to extract the set of trajectory segments from x , each equally long as n_q (the length of q) and starting at the indices i for $i \in 1 \dots n_x - n_q + 1$. Then we calculate the distance δ^f of all these segments to q . In a brute force approach we calculate all these distances independently from each other, so we need to get the parameters to shift, scale and rotate the segments independently from previous distance calculations, even though, there might be a relation of the parameters for consecutive segments. Also, the naive approach does not apply early abandoning to reduce the number of computed operations. The brute force approach would require too much computation time for applying the method on big data sets or in a real time scenario where the analysis needs to be completed before new data is recorded. A more efficient algorithm to calculate $s3kr(q, x)$ would enable a broader applicability.

We propose the algorithms S3KR (described in detail in Algorithm 1) and DELTA (described in Algorithm 2) to calculate $s3kr$ by efficiently making use of the relations of the parameters for scale, shift and rotation. The procedure DELTA is called inside the algorithm S3KR with the rotation matrix R and scaling factor s as input parameters, which are updated inside of the S3KR algorithm. Algorithm 2 can calculate both δ and δ^f , dependent on the input parameter R . The former is achieved if R rotates x to fit q , and the latter if R rotates x to fit the reference f . q is supposed to be already scaled and shifted (and rotated to fit f), see Line 3 in Algorithm 1. In the following we walk through the algorithms in detail and discuss how we apply the following strategies for the respective components of the algorithms to accelerate the calculation of δ^f and $s3kr$:

- (a) Incremental update of the dimension means and variances,
- (b) Selection of a reference f ,
- (c) Replacement of the SVD by the exact expression of the singular values (for $m = 2$),
- (d) Early abandoning and decoupling of the calculation of H and R (see (3)) from the actually scaled and shifted time series.

Figure 3 shows results of runtime experiments², where we compare the adjusted components of the algorithm S3KR to the brute force computation. We simulate synthetic trajectories of variable lengths, and q is simulated to be of equal length as $n_x/10$.

Algorithm 1 Calculate the distance vector $s3kr(y, q)$.

```

1: function S3KR( $y \in \mathbb{R}^{n_y \times m}$ ,  $q \in \mathbb{R}^{n_q \times m}$ ,  $\epsilon$ )
2:    $d \leftarrow$  empty vector
3:    $q \leftarrow \pi^f(\gamma(\tau(q)))$ 
4:   initially calculate  $\mu$  and  $s$  ▷ standard formula
5:   set initially  $H$  and  $R$  ▷ see (14, 3)
6:    $d_1 \leftarrow$  DELTA( $\{y\}_{1, n_q}, q, R, s, \epsilon$ )
7:   for  $i = 2 : n_y - n_q + 1$  do
8:     update  $\mu$  and  $s$  ▷ see (12)
9:     update  $H$  and  $R$  ▷ see (14, 3)
10:     $d_i \leftarrow$  DELTA( $\{y\}_{i, n_q}, q, R, s, \epsilon$ )
11:  end for
12:  return  $d$ 
13: end function

```

- a) **Incremental update:** The sliding window algorithm $s3kr$ benefits from incrementally updating the scaling factor for γ . This is possible since the scaling factor is a function of

² We used a standard laptop computer with 2.8 GHz and 16GB RAM for the runtime comparisons.

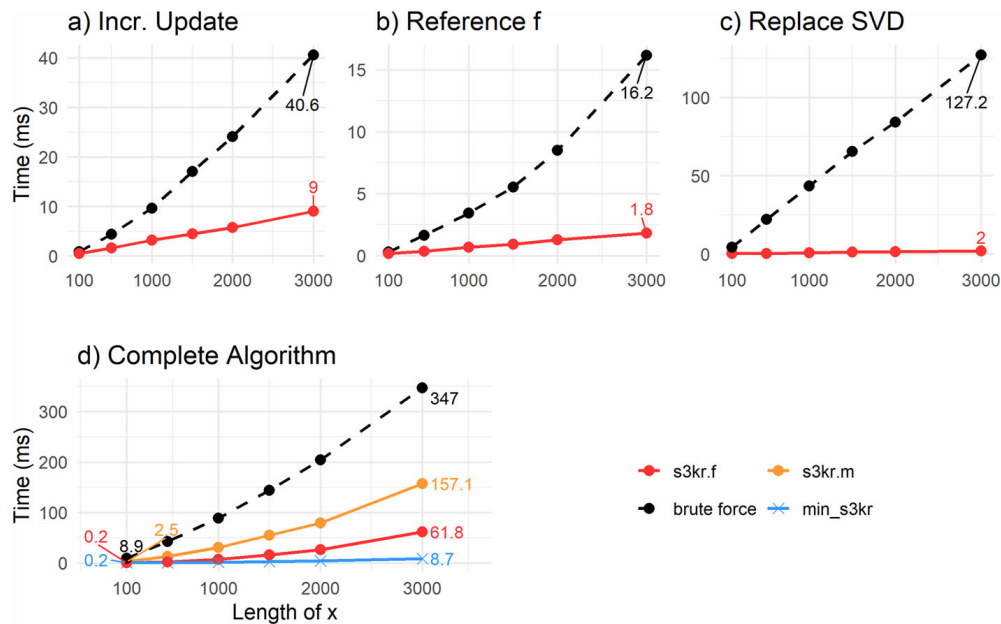


Fig. 3 Runtime comparisons for S3KR: (a) to (c) show runtimes of algorithm components of S3KR relative to the brute force counterparts. (d) shows absolute runtimes for the complete algorithm S3KR, the brute force search counterpart and the 1-NN search via min_s3kr. (a): Incremental update of μ and σ versus calculation from scratch for each time step (brute force). (b): Calculation of H by cross product (brute force) or fast update due to f . (c): Get R by the exact expression of the singular values versus the SVD (brute force)

Algorithm 2 Calculate the distance $\delta(x, q)$ with scaling, shifting and rotating inside the distance computation.

```

1: procedure DELTA( $x, q \in \mathbb{R}^{n \times m}, R \in \mathbb{R}^{m \times m}, s, \epsilon$ )
2:    $\delta \leftarrow 0$ 
3:   for  $i = n : 1$  do
4:      $d \leftarrow s(x_i - x_1) \cdot R - q_i$ 
5:      $\delta \leftarrow \delta + \sqrt{\sum d^2}$ 
6:     if  $\delta > \epsilon$  then return NaN
7:   end for
8:   return  $\delta$ 
9: end procedure
    
```

▷ reverse order
 ▷ shift, scale and rotate inside the for-loop
 ▷ d is a row vector, δ is a scalar
 ▷ early abandon if δ hits ϵ

the standard deviations per dimension, σ_j . Having $\sigma_{i,j}$, then $\sigma_{i+1,j}$ is the square root of $\sigma_{i+1,j}^2$:

$$\mu_{i+1,j} = \mu_{i,j} + \frac{(x_{i+n,j} - x_{i,j})}{n}$$

$$\sigma_{i+1,j}^2 = \sigma_{i,j}^2 + \frac{(x_{i+n,j}^2 - x_{i,j}^2)}{n-1} + (\mu_{i,j}^2 - \mu_{i+1,j}^2) \frac{n}{n-1}. \tag{12}$$

Figure 3a demonstrates that incrementally updating (red line) μ and σ is up to four times faster than the brute force method (dashed black).

- b) **Selecting a reference f :** To demonstrate the benefit of f , we start with rotating the scaled and shifted version of y to fit f . We need the matrix H , where $H = f^T \cdot \gamma(\tau_0(y))$ (see (3)):

$$H_{ij} = \sum_{k=1}^n x_{ki} \cdot (\gamma(\tau_0(y)))_{kj} = \underbrace{\left(\sum_{l=1}^m \sigma_l^2 \right)^{-1/2}}_s \left(\sum_{k=1}^n f_{ki} \cdot y_{kj} - y_{1j} \cdot \sum_{k=1}^n f_{ki} \right) \tag{13}$$

$$\Rightarrow H = s(f^T y - (f^T \mathbf{1})y_1),$$

where³ $\mathbf{1}$ is the column vector of ones and y_1 is the row vector of the first row of y , so the observation at time $i = 1$. Next, we set f equal the matrix of zeros in all but the last dimension, and ones in the last dimension. For $m = 2$, (13) reduces to

$$H = s \cdot n_q \begin{bmatrix} 0 & 0 \\ \mu_{i,1} - y_{i1} & \mu_{i,2} - y_{i2} \end{bmatrix} \tag{14}$$

where $\mu_{i,1}$ is the mean of all $y_{j1} \forall i \leq j \leq i + n_q - 1$, and μ is updated incrementally via (12). Calculating H via (14) instead of (13) saves time since no cross-products of q and the segment of y are necessary, so reduces the runtime complexity for updating H from $O(n_q)$ to $O(1)$ per sliding step. Figure 3b shows the actual effect on the algorithm by comparing the runtime for the cross product (black dashed) to the update of H via (14) (in red). Since the number of sliding steps increase with the length of x (x-axis), the red line shows linear increase even though the complexity of a single sliding step is $O(1)$.

- c) **Replace the SVD:** For the popular case of $m = 2$ (e.g. 2-dimensional GPS trajectories) we derive the rotation angle as described in Section 4.6 of [5], by making use of the exact expression of the singular values of H in (3). Further we make use of the function *atan2* to return unambiguous rotation angles for the full range of 0 to 360 degrees.

$$\alpha = \text{atan2}(H_{1,2} - H_{2,1}, H_{1,1} + H_{2,2})$$

$$R = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix}. \tag{15}$$

Figure 3c illustrates that in our experiments this calculation of R is up to 60 times faster than the brute force method which derives R according to (7).

- d) **Early abandoning:** We come back to the example at the beginning of Section 3.3 and ask the slightly adjusted question, where to find in Fig. 1 a) all segments having a distance smaller than a given threshold ϵ to q (an ϵ -query), or similarly b) just the segment with the smallest distance (the 1-nearest neighbor search, 1-NN). Again, the brute force approach is to calculate all distances, and in a second step, for a) to decide which distances are smaller than ϵ , and for b) to apply a linear search to find the minimum. To answer the ϵ -query and 1-NN search efficiently, we propose the algorithm S3KR (Alg. 3), that saves computation time by abandoning as much as possible of the calculation process of distances larger than ϵ . Section 3.2 already discussed early abandoning (see (11) and the accompanying example) and next we discuss how Algs. 2 and 3 apply early abandoning.

For calculating the distance of q and a segment of $y - \delta^f(q, \{y\}_{i,n_q})$ – we need to shift, scale and rotate only the segment of y , since $\pi^f(\gamma(\tau_0(q)))$ is performed only once $\forall i$ (see Line 3 of Alg. 1). The rotation matrix R depends on H , which is the cross

³ For calculating $\delta(q, y)$ – instead of $\delta^f(q, y)$ – the derivation of H is analogous to (13), but we substitute f by q , and y_1 by the vector of column means.

product of f and $\gamma(\tau_0(y))$ (see (3)). However, (14) showed that we can express H as a function of f , y and s , but independent of $\tau(\gamma(y))$. Consequently we can calculate $R^* = \operatorname{argmin}_R d_E(f, \tau(\gamma(y)) \cdot R)$ before actually scaling and shifting y . Only the scaling factor s is required, which is updated incrementally at $O(1)$ costs, see (12). This is necessary to plug in the operators τ , γ and π^f directly into the implementation of the distance calculation, implemented as for loop iterating over the time index (see Line 4, Alg. 2). As soon as the accumulated distance hits the threshold ϵ , the for loop breaks and further unnecessary scaling, shifting and rotation calculations are abandoned. This way of decoupling the calculation of the rotation matrix, and scaling and shifting the time series helps the algorithm S3KR to save computation time, since unnecessary shifting, scaling and rotation operations are abandoned.

With the help of early abandoning of DELTA inside of S3KR we can considerably accelerate k-NN searches and ϵ -queries, where all distances smaller than ϵ are returned. Algorithm 3 details MIN_S3KR, which is the algorithm to calculate $\min_s3kr()$, so to answer the 1-NN search. The algorithm is similar to S3KR, but updates the threshold ϵ dependent on the best so far detected index (see Line 12, Alg. 3). With minor adaptations it can also answer the k-NN query: Adjust the threshold management with regard to the latest (to guarantee no overlap) and k best so far detected distances. For details about this threshold management we refer to [10].

Figure 3d compares the absolute runtimes of S3KR (red solid) and a brute force method (black dashed) which does not take advantage of the discussed acceleration methods a)-c). S3KR is 6 to 45 times faster than the brute force method. The figure also shows runtimes for the 1-NN search by MIN_S3KR that benefits evidently from early abandoning, and needs less than 10 ms for the nearest neighbor search with $n_x = 3000$.

Algorithm 3 Calculate $\min_s3kr(x, q)$ and the location where to find it.

```

1: function MIN_S3KR( $y \in \mathbb{R}^{n_y \times m}$ ,  $q \in \mathbb{R}^{n_q \times m}$ )
2:   ( $d^*$ ,  $i^*$ ,  $\epsilon^*$ )  $\leftarrow$  (NaN, NaN,  $\epsilon$ )
3:    $q \leftarrow \pi^f(\gamma(\tau(q)))$ 
4:   initially calculate  $\mu$  and  $s$  ▷ standard formula
5:   set initially  $H$  and  $R$  ▷ see (14, 3)
6:    $d \leftarrow \text{DELTA}(\{y\}_{1,n_q}, q, R, s, \epsilon)$ 
7:   if  $d \neq \text{NaN}$  then ( $d^*$ ,  $i^*$ ,  $\epsilon^*$ )  $\leftarrow$  ( $d$ ,  $i$ ,  $d$ ) ▷  $\epsilon^* \leq \epsilon$ 
8:   for  $i = 2 : n_y - n_q + 1$  do
9:     update  $\mu$  and  $s$  ▷ see (12)
10:    update  $H$  and  $R$  ▷ see (14, 3)
11:     $d \leftarrow \text{DELTA}(\{y\}_{i,n_q}, q, R, s, \epsilon^*)$ 
12:    if  $d \neq \text{NaN}$  then ( $d^*$ ,  $i^*$ ,  $\epsilon^*$ )  $\leftarrow$  ( $d$ ,  $i$ ,  $d$ )
13:  end for
14:  return ( $d^*$ ,  $i^*$ )
15: end function

```

3.4 TSS clustering with s3kr

Equipped with the presented sliding algorithm S3KR, we propose a clustering algorithm that applies S3KR to find clusters of shift-scaling-rotation invariant segments of a trajectory.

Figure 4a) motivates this by showing a trajectory y forming two spirals. To reveal structure and repetitive patterns in y we want to cluster the segments of y . But, when we want to cluster the overlapping time series subsequences (TSS) of the trajectory y , the clustering easily becomes meaningless (as discussed in [7]), since the closest neighbor of a segment is often the trivial neighbor with a high overlap, starting one time index earlier or later. Hence, we propose an algorithm that avoids these pitfalls by setting distances of segments with an overlap equal infinity. This ensures that overlapping segments can only end up in the same cluster, if they are both close enough to a third segment, that has no overlap with either of the two. To avoid these pitfalls of calculating meaningless cluster representatives due to overlapping subsequences, we propose the following algorithm to cluster a time series y of length m , with a subsequence length n and number of clusters k :

1. Distance Matrix: Apply s3kr to calculate the distance matrix D of all pairwise distances δ^f of all segments of y . So we get $D_{ij} = \delta^f(\{y\}_{i,n}, \{y\}_{j,n})$.
2. Set $D_{ij} = \infty$ where $|i - j| < n$. D now represents a weighted graph where all subsequences are connected, except they have an overlap.
3. Perform Spectral Clustering [23]: Two overlapping segments can only end in the same cluster, if they are close enough to a third segment that does not overlap with either. Consider the segments of y in Fig. 4c) (here $m = 1000$ and $n = 30$) that start at the first and second index. Both describe the movement of a left turn, but $D_{1,2}$ is set to ∞ . Since both have a small distance to segments starting after index 31, they end up in the same cluster, describing a left turn.
4. Get cluster representatives: Say a cluster c consists of segments starting at the indices $i \in I^c$. According to step 3, a cluster can contain overlapping segments. However, the representation of a cluster benefits from presenting only non-overlapping segments, instead of showing redundant information. We define the degree vector for a cluster c , $g^c \in \mathbf{R}^{m-n+1}$ for $l \in (1, \dots, m - n + 1)$ as:

$$g_l^c = \begin{cases} \infty, & \text{if } l \notin I^c. \\ \frac{1}{\text{number of elements in } I^c, \text{ where } |l-j| \geq n} \sum_{\substack{j \in I^c \\ |l-j| \geq n}} D_{lj}, & \text{if } l \in I^c. \end{cases} \quad (16)$$

So, g_l^c is the mean of the distances of the segment starting at index l , $y_{l,n}$, to all segments in the same cluster c , $y_{j,n}$, where $j \in I^c$, and the segments $y_{l,n}$ and $y_{j,n}$ have no overlap. Then g^c is the vector of average cluster-internal distances. We call those segments, that

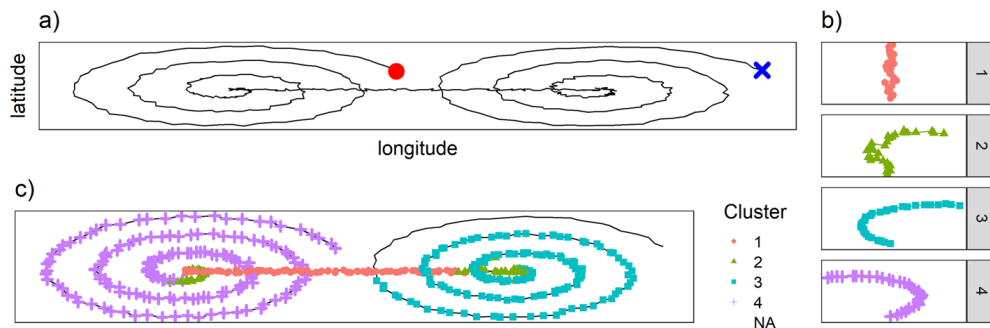


Fig. 4 a) A simulated trajectory starting at the red dot, ending at the blue cross. b) The cluster centers after applying TSS clustering. c) The observations are colored according to the assigned clusters of the segments starting at these observations

represent a local minima of g^c the characteristic segments of c , because these segments represent the cluster the best. In detail, a segment $y_{l,n}$ is a characteristic segment (seg^{char}) of cluster c , if $g_l^c = \min_j(g_j^c) \forall |l - j| < n$. The condition $|l - j| < n$ guarantees that characteristic segments do not overlap.

It is sometimes desirable to have a single representative time series for each cluster. In such cases, we average the seg^{char} of each cluster. Figure 4b) depicts the average of the seg^{char} for each cluster in that example, and together with Fig. 4c) emphasizes that the two spirals look identical at first glance, but are of different directions, and so assigned to the clusters 3 and 4. The first cluster describes the straight connection line of the two spirals, and the second cluster represents the transition between the others.

With the spectral approach, our algorithm assigns segments that are not directly connected to a common cluster if sufficient indirect connections via other segments exist. A density-based clustering algorithm such as DBSCAN would achieve a similar result, indeed there are strong theoretical relationships between spectral and density-based methods [17]. However, unlike centroid-based methods, DBSCAN does not detect representative points for each cluster. We obtain representative points from the degree distribution of the similarity matrix. Those representative points greatly facilitate the interpretation of the result.

To cluster the subsequences of a set of time series we apply the former algorithm on each single time series and collect all the seg^{char} of all trips, to clusters these again with spectral clustering. Again, if desired, we can average the segments without any concern of overlap, since the seg^{char} are defined to have no overlap.

Apart from the example in Fig. 4, we also apply this TSS clustering, (a) on a single GPS time series in Section 5.1, and (b) on a set of GPS time series in Section 5.2, and demonstrate that it detects meaningful clusters and cluster centers in both applications.

4 Experimental results

Section 3 elaborates the rotation-invariant distance metric δ^f , the sliding algorithm S3KR and finally the TSS algorithm, which build on each other. Before we apply δ^f embedded in S3KR or TSS, we test the key component δ^f itself. In detail, the experiments in the following sections demonstrate the following:

- Sections 4.1 & 4.2: The procrustes distances are rotation-invariant, more robust to noise, and faster than the methods from literature.
- Section 4.3: Since δ^f is a metric, we can build an M-Tree data structure with δ^f as distance metric. For a nearest neighbor search in a set of trajectories of equal lengths, the M-Tree outperforms the brute force search.
- Section 4.4: Applied on a benchmark dataset of trajectories of varying lengths, the proposed sliding algorithm s3kr performs better, in terms of accuracy and runtime, compared to the methods from literature.

Sections 4.1 and 4.2 evaluate and demonstrate the capabilities of the procrustes distances in an isolated setting, whereas the Sections 4.3, 4.4 and 5 demonstrate the procrustes distances embedded in an M-Tree, a sliding algorithm and a clustering algorithm.

In Section 4.1, 4.2 and 4.3 we simulate trajectories. In Section 4.4 we make use of an open source benchmark data set. The case studies in Sections 5.1 and 5.2 demonstrate how to apply our methods on real world (maritime and urban) GPS trajectories. We selected this elaborate set of experiments to demonstrate the benefits of our proposed methods.

Table 2 Overview of the parameters used in the experiments and case studies presented in the various sections

	Section 4.1	Section 4.2	Section 4.3	Section 4.4	Section 5.1	Section 5.2
k	3				4	9
σ	{0, 0.2, 0.4, 0.6, 0.8, 1}		1			
ws	{0, 5, 10, 20, <i>None</i> } in %	ws^*	ws^*	ws^*		

k is the number of clusters for the clustering algorithm. σ is the standard deviation of added noise for simulating synthetic trajectories. ws is the Sake Chiba [15] window size parameter for the DTW algorithm. ws^* is the set of ws values, which show best performance for the benchmark methods in Section 4.1. The respective sections provide more details about the choice of the parameters

We compare our proposed methods with the following methods from literature applied on trajectory data or time series data.

z-norm: The z-normalization (as e.g. applied in [12, 16]) combined with the Euclidean distance d_E or Dynamic Time Warping (DTW) is not rotation invariant. However, here it deals as baseline method to point out the challenges of discovering rotation invariant patterns in trajectories.

arc-angle: The projection of the coordinates of a trajectory into the space of arc-length and angle ([3]) is a typical approach to measure shift and rotation independent similarities. In the sense of a fair comparison we include a normalization step (according to (2)) to make it scale invariant.

shape-sgnt: The shape signature is a 1-dimensional representation of 2-dimensional shapes, by measuring distances of all points of the shape to its center. This method is often applied in shape recognition [24], and is rotation and shift invariant. We z-normalise the 1-dimensional representation to achieve scale invariance.

cMass: The work by [19] presents a shift-scale-rotation invariant distance measure for trajectories, that transfers the 2-dimensional longitude-latitude time series into the arc-angle space, and then interpolates the angles to be equidistant sampled, and uses DTW to measure the distance. This work proposes three different methods to calculate the angles: 'Exact', 'Relative' and 'cMass'. We evaluated all three of these in our experiments in Section 4.1 with varying window size parameters for DTW. Since cMass outperforms the other two by far, and also performs best in the original work [19], we compare our methods with cMass in the remaining experiments. We set the maximum number of iterations for the modulo normalisation algorithm (described in [19]) equal 10 for the runtime experiments, and equal 100 for the remaining experiments.

We combine the methods z-norm, arc-ang, shape-sgnt and all three variations of [19] with d_E and DTW – with the Sakoe Chiba window [15] of 5, 10, 25 and 100% of the time series length. It is worth noting, that DTW with a window size parameter equal 0, is equivalent to applying the d_E . Our initial experiment in Section 4.1 analyses the performance of all these methods dependent on the window size parameter. In the remaining experiments we apply the window sizes that showed best performance in the initial experiment. Table 2 gives an overview of the used parameters across all the experiments in this section and the case studies in Section 5.

We⁴ used a standard laptop computer with 2.8 GHz and 16GB RAM for the runtime comparisons. For a fast DTW computation we applied the R package IncDTW [9], and for cMass we adjusted the local distance and lower bound method as described in [19].

⁴ To reproduce our results presented in Sections 4 and 5 all data and code are here: <https://tinyurl.com/d7jrestk>

4.1 Clustering synthetic random walks

We simulate three 2-dimensional random walks of 50 observations as cluster centers. We copy each of the cluster centers 20 times and randomly modify the copies by adding white noise, scaling by a random positive factor, shifting by a random 2-dimensional vector and rotating by a random angle between 0 and 360 degrees. Appendix hyperlinkappen1A gives further details about the simulation of random trajectories. Then we calculate the pairwise distance matrices and cluster these with PAM [14] (with $k = 3$, so searching for 3 centers). To keep it straight forward, we set k equal to the number of clusters we simulated and focus the presentation on the Procrustes distance performance. For each combination of the standard

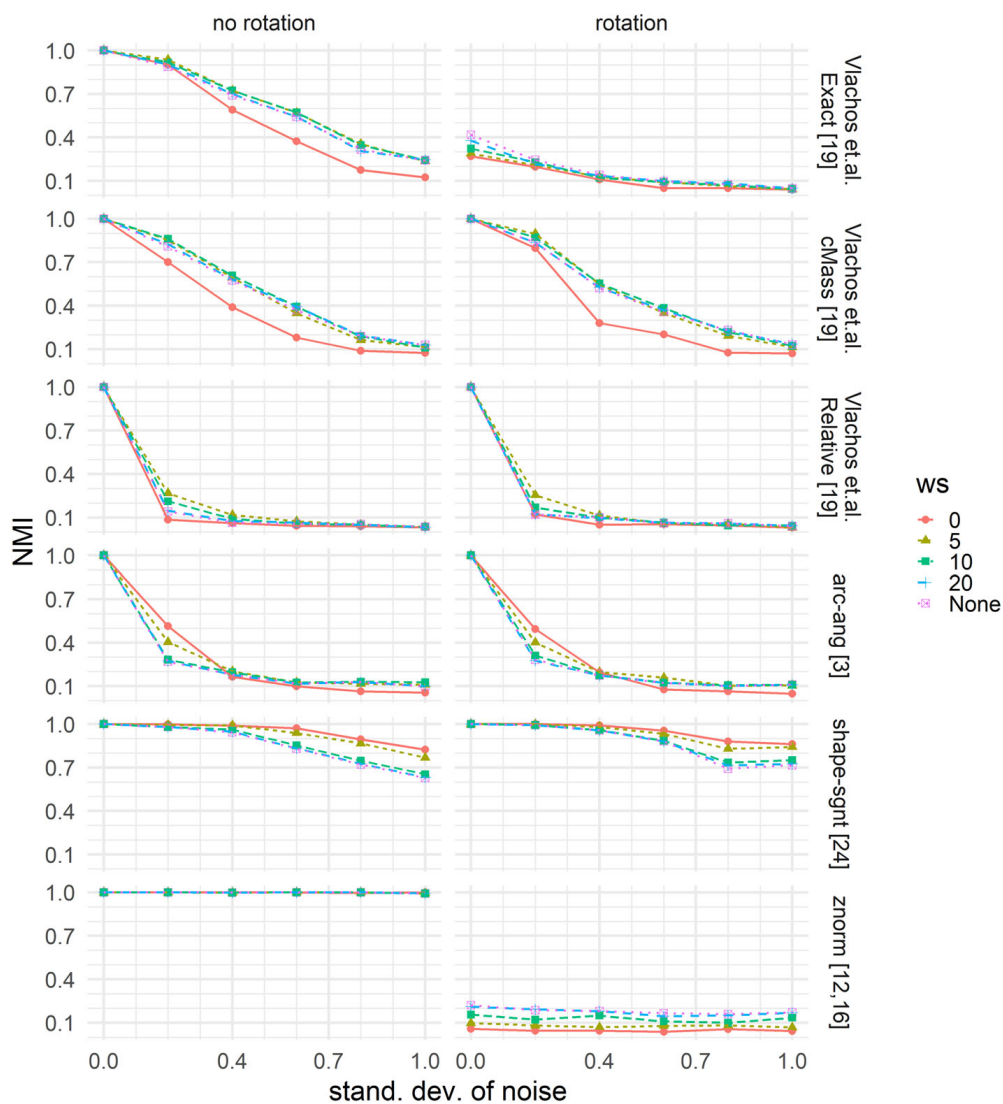


Fig. 5 Clustering performance: The baseline methods are parameterised with varying values of ws and applied on synthetic trajectories. The rotation angle for simulating the time series is either constant 0 (left) or sampled between 0 and 360 degrees (right)

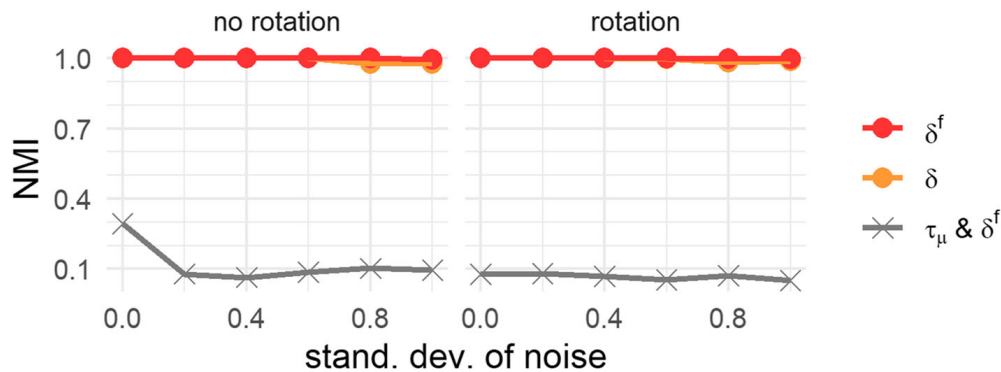


Fig. 6 Comparison of the proposed distance function with the variation τ_μ & δ^f

deviation of the added noise, and whether random walks are rotated or not, we repeat the whole experiment 30 times to exclude the possibility of a method's superiority by chance.

We compare our proposed methods with baseline methods that rely on DTW. The performance, as well as the runtime of DTW heavily depend on the window size parameter (ws). When comparing two time series x and y , ws defines the maximum number of observations⁵ that can be matched from x to y .

To guarantee a thorough and fair comparison with the baseline methods, we first evaluate for what values of ws the baseline methods perform best. Figure 5 plots the normalized mutual information (NMI)s versus the standard deviation of added white noise, for rotated (right) and not-rotated (left) random walks. Since we are interested in detecting rotation invariant similarities, we select the best-in-class based on the performance on the right hand side of Fig. 5 (cMass with $ws = 10$, arc-ang with $ws = 5$, znorm with $ws = None$, shape-sgmt with $ws = 0$ so the Euclidean distance). For the rest of this work we apply the methods with these values for ws .

We also tested δ^f in combination with τ_μ . Figure 6 depicts, that this variation hardly identifies any structure in the data. We concluded that the rotation towards a reference time series that starts in the origin only shows meaningful results when both trajectories also start at the origin, which τ_0 guarantees, but τ_μ does not.

Finally, Fig. 7 presents the comparison of our proposed methods with the baseline methods, where the applied ws parameters are the best-in-class from Fig. 5

For a deeper visual inspection of the performance of the distance measures in Fig. 7 – independent from a clustering algorithm –, we also plot the pairwise distance matrices. Figure 8 shows the pairwise distance matrices for all trajectories from one of the 30 repetitions, where the trajectories were rotated randomly and noise was added with a standard deviation of 0.6. One can easily identify the clear structure of the three clusters for δ^f and δ , but hardly for the shape-signature or one of the others.

This experiment demonstrates a) the Procrustes distances seem to be more robust to noise and clearly outperform the others, and b) that traditional time series distance measures that ignore the rotation of trajectories are incapable of measuring similarities independent of rotations.

⁵ To be precise, with $ws = 5$ we note that the window size is equal 5% of the length of x , $ws = 0$ is identical to calculating the Euclidean distance, and a $ws = 'None'$ means that the alignment of x and y is not constrained by a window at all. We refer to [15] or [10] for a detailed discussion of ws .

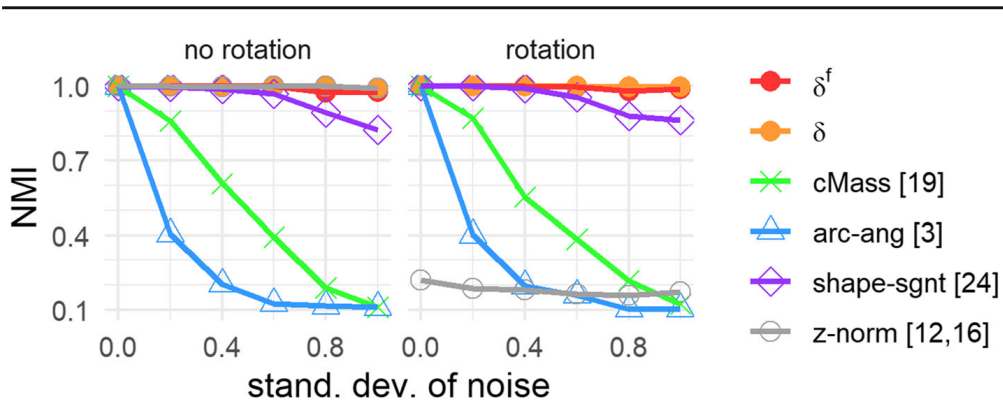


Fig. 7 Amount of white noise versus NMIs for clustering synthetic time series based on different distance measures

4.2 Runtime comparison

Section 3.3 discusses via Fig. 3d the runtimes for the sliding algorithm S3KR (with δ^f , or δ), and compares these with the brute force alternative and MIN_S3KR. Here we complement these experiments and compare the runtimes for distance measures only, i.e. without sliding algorithm. We simulate 2-dimensional random walks of equal lengths, ranging from 50 to 1000, and apply the same methods which show best results in Section 4.1, especially in connection with the Sakoe Chiba window size parameter for DTW. Figure 9 shows that δ^f and δ are of similar speed, and outperform the other methods by a factor of 10 to 470.

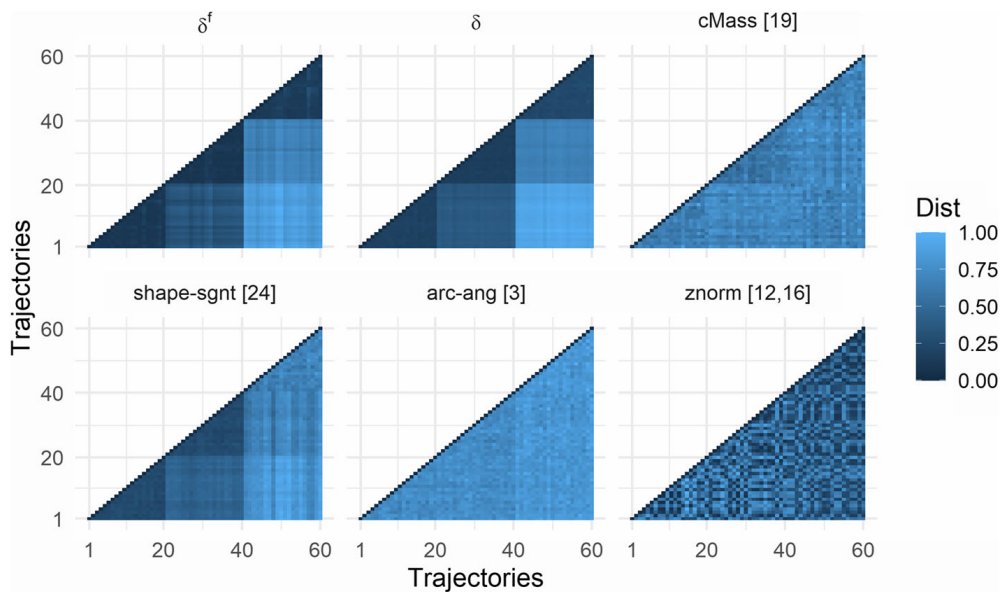


Fig. 8 Pairwise distance matrices for a simulated dataset of 3 clusters of 2-dimensional random walks. Due to the design of the experiment, the first 20 trajectories are part of the first cluster, the trajectories 21 to 40 belong to cluster 2, and the remaining to the third cluster

Geoinformatica

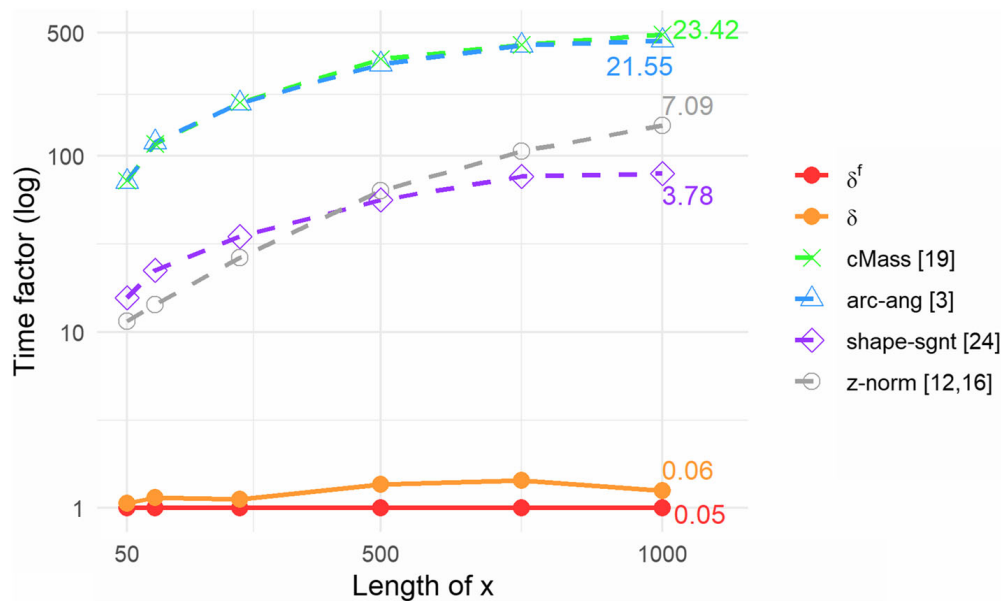


Fig. 9 Computation runtimes given on the y-axis as factor relative to the fastest method, δ^f . And the absolute computation times in ms printed at the very right, for a length of x equal 1000

4.3 Classifying synthetic random walks with an M-Tree

One main advantage of our proposed distance measure δ^f is, that it is a metric, and so fulfills the triangle inequality. In this experiment we demonstrate how we can make use of this beneficial characteristic. Similar to the previous example, we simulate 2-dimensional random walks (1000 random walks, each of length 50 observations, standard deviation of noise = 1, belonging to 10 different groups) and classify these with a 1-NN classifier, in a 10-fold cross validation. The results in Table 3 demonstrate that the Procrustes distances outperform the others in accuracy and runtime. But most importantly, we can also fulfill this task by building an M-Tree [2], which is a special data structure for range queries and nearest neighbor searches. Building an M-Tree⁶ requires as input a set of observations (in our experiment this is the set of all the trajectories from 9 out of the 10 folds, so 900 trajectories) and a distance function fulfilling the triangle inequality, δ^f . This allows to perform a nearest neighbor search without the need of comparison with all data in the tree. We set the maximum node size of the M-Tree equal 3. To classify a single trajectory takes on average only 71 distance calculations to step down the tree, and find the nearest neighbor. In comparison, without the M-Tree (as performed for all but the first column in Table 3) we need to calculate the distances to all trajectories in the train data, so 900. So we save lots of computation time, which is also why the runtime for the combination δ^f & M-Tree is more than 10 times faster than for δ . It is worth mentioning that for each distance calculation only one rotation is necessary (compare Eq. 8), since the time series spanning the tree are already rotated to fit f . Building the M-Tree involves a certain amount of work, since it requires multiple distance calculations to setup this data structure. In this experiment we measured 4.4 seconds to set up the M-Tree for one of the 10 folds, meaning a database of 900 trajectories. However, this overhead is negligible for classifying a trajectory based on an already existing M-Tree. We

⁶ We applied the implementation of <https://github.com/tburette/mtree> which follows the original paper [2]

Table 3 1st row: Average F1 scores (%) for classifying the trajectories with a 1-NN classifier and 10-fold cross validation. 2nd row: Average runtime (in milliseconds) for calculating all distances required to classify a single trajectory

	δ^f & M-Tree	δ^f	δ	cMass [19]	z-norm [12, 16]	shape-sgnt [24]	arc-ang [3]
F1	0.98	0.98	0.97	0.25	0.85	0.84	0.26
runtime (ms)	20	225	218	3757	574	669	3248

The bold entries emphasise the best results in the presented comparison

conclude, that δ^f shows a prediction accuracy just as good, or even slightly better than δ , and moreover that being a metric is a major advantage for performing a range query or nearest neighbor search most efficiently.

4.4 Recognizing handwritten symbols

To compare our methods with the distance measures proposed by [19] on a benchmark trajectory data set, we downloaded time series data from the UCI⁷ repository consisting of 2858 trajectories describing the movement of a pen when writing 20 different symbols [21]. These trajectories are similar to traditional GPS trajectories. The trajectories consist of the two dimensions longitude and latitude, and are of different lengths, ranging from 60 to 182 observations. We built a 10-fold cross validation 1-NN classifier based on the same distance measures as in the previous experiment on synthetic data (Section 4.1), but in a sliding fashion and take the minimum of all distances if the two compared time series differ in length. Table 4 summarizes the F1 scores and shows that our proposed Procrustes distance with fixed reference f and zero-shift operator outperforms the others.

We also measured the computation times for calculating all required distances in this experiment. The second row of Table 4 shows that our methods outperform the methods from literature in terms of runtime by a factor of 1.6 to 100.

The problem statement of this experiment differs slightly from the one in Section 4.3. The trajectories in Section 4.3 are all of the same length, which they are not in this experiment. Consequently, here we apply another distance measure: Two trajectories are similar, if we find a good match of the shorter trajectory in the longer trajectory. For this reason we apply `min_s3kr`. This experiment showcases the sliding algorithm, and that it is capable of detecting similar patterns.

Applying an M-Tree here would be possible, however with some complications: Consider two trajectories x and y , and say x is longer than y , so $n_x > n_y$. For the comparison of x and y we would need to build an M-Tree (we notate this M-Tree as M_{xy}) out of all the segments of x of length n_y , $\{x\}_{..n_y}$. Similar to Section 4.3 this would require the overhead costs of building M_{xy} , before finding the most similar segment of x to y , and their distance. Contrary to Section 4.3, here the overhead costs would not be negligible, because in general M_{xy} is of no further usage for comparing x with other trajectories. For comparing x to another trajectory z , a new M-Tree M_{xz} would be required out of the segments $\{x\}_{..n_z}$ if $n_x > n_z$, or out of the segments $\{z\}_{..n_x}$ if $n_x < n_z$. Only if $n_x = n_y$ (which is unlikely for real world GPS trajectories), the former M-Tree M_{xy} could be reused for comparing x and z .

⁷ Dua, D., Graff, C.: UCI ML repository (2017), <https://archive.ics.uci.edu/ml/datasets/Character+Trajectories>

Table 4 1st row: Average F1 scores (%) for classifying the handwritten symbols with a 1-NN classifier and 10-fold cross validation. 2nd row: Average runtime (in milliseconds) for calculating all distances required in a sliding algorithm for comparing two time series. The algorithm MIN_S3KR is applied to compute the distances δ^f and δ

	δ^f	δ	cMass [19]	z-norm [12],[16]	shape-sgnt. [24]	arc-ang. [3]
F1 (%)	89.40	81.63	80.02	57.35	31.67	36.80
Runtime (ms)	0.3	0.4	5.8	0.5	6.2	30.6

The bold entries emphasise the best results in the presented comparison



(a) GPS record, starting at the green marker and ending at purple. The zoomed area (right) shows the ship at dredging. (b) Cluster means (red) and char. segments (black) of the trajectory. One single right turn (orange dashed) in C3.

Fig. 10 Dredger ship's GPS trajectory and the clusters describing the ship's movements. The clusters C1 to C4 consist of 32, 18, 15 and 3 seg^{char} , respectively

We conclude, that there are use cases for both approaches (M-Tree, and sliding algorithm), and that our proposed methods can help to efficiently perform rotation invariant pattern recognition in both situations.

5 Case studies

The following two case studies demonstrate how the TSS algorithm helps to cluster GPS trajectories and detect clusters of similar (rotation-invariant) trajectory segments. These clusters help to gain higher level knowledge and better understand the data.

5.1 Nautical GPS trajectories

To showcase the algorithm for clustering time series subsequences (see 3.4), we analyzed historical GPS tracks from the AIS database⁸. Figure 10 shows an approximately 50km extract of a GPS track of a dredger ship in front of the Danish coast, northwestern of Copenhagen. The record (shown in the left part) starts at the green marker, and ends at the purple marker. The right part of Fig. 10 zooms in the area where the ship is dredging.

We applied the TSS clustering algorithm from Section 3.4 on the approx. 2000 overlapping segments of 500 meters length each. We set k (as we also did in Section 5.2) by visual inspection to get the full spectrum of distinct movement patterns in the data. Figure 10 shows that (a) the biggest cluster 1 describes a straight pattern and is mainly formed by the ship going to the purple marker after finishing the dredging, (b) clusters 2 and 3 describe a left curve and a left u-turn, (c) cluster 4 is small and consists of only $N=3$ seg^{char} , (d) the ship does only one hard right turn, which is the dashed orange line in cluster 3.

The clustering presented in Fig. 10 achieves a value of 92% for the share of the between-sum-of-squares divided by the total-sum-of-squares ($\frac{bss}{tss}$). This means a considerable amount of the total variation in the data is explained by the clustering.

Applying the TSS algorithm requires the calculation of many time series distances, best performed in a sliding approach, as the algorithm S3KR proposes. To point out the benefit of S3KR compared to a brute force approach, we performed this experiment with both approaches and measured the computation time. The calculated distances are identical. S3KR needs 0.8 seconds, whereas the brute force approach needs 51.4 seconds, so S3KR is about 65 times faster. This points out the major benefit of S3KR compared to a naive method, especially for analysing big data bases of GPS trajectory data.

⁸ AIS automatic identification system, <https://www.dma.dk>. Contains data from the Danish Maritime Authority that is used in accordance with the conditions for the use of Danish public data.

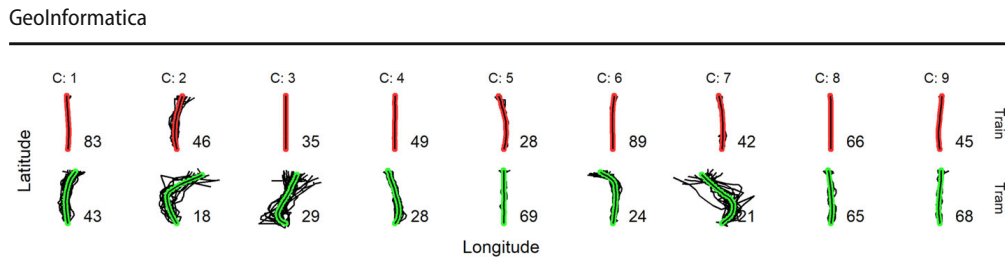


Fig. 11 Cluster centers for train (upper, red) and tram (lower, green), starting at the bottom, next to the numbers of characteristic segments forming that cluster

5.2 Urban GPS trajectories

In the final experiment we demonstrate the algorithm for clustering time series subsequences of a set of time series (see Section 3.4). We recorded GPS tracks while travelling by train (6 trips) and tram/light-rail (9 trips) in Vienna, Austria and Zurich, Switzerland. These 15 trips have a total length of ca. 185km, split into 18200 segments of 200m each. We aim to discover distinguished maneuver patterns, rather than splitting the data by the transport modes. That is why we clustered all the train trips and tram trips separately. Figure 11 compares the results. For both modes of transport we set $k = 9$. The upper row shows the red cluster centers for train, and the lower row in green for tram. As expected some of the clusters are characterized by straight movements. The prototypical patterns for tram show more variations for turning movements, especially show angles of higher degrees than for the train. This is reasonable since the rails for trains are typically shaped differently to those for trams, that are embedded in the road network. As in Section 5.1 we also measured the ratios $\frac{bss}{tss}$ for the clustering results in Fig. 11, which are 81% for train and 76% for tram. This is just inline with what the visual inspection of Fig. 11 shows. We see slightly more variation within some of the clusters for tram, which means the within-sum-of-squares $wss = tss - bss$ is higher for tram, consequently less of the total variation is explained by the clustering, and $\frac{bss}{tss}$ is smaller for tram than for train. The results of this experiment can help a transport mode classifier model to distinguish transport modes based on recognizing transport mode specific driving maneuvers.

6 Related work

The unique characteristic of our method is to detect scale-shift-rotation invariant similarities of subsequences of trajectories. Even though much work has been developed for trajectory analysis [25], only few discuss rotation invariance, and to the best of our knowledge none in combination with an algorithm for sliding pattern recognition (1-NN and ϵ -queries), or subsequence clustering. In Section 4 we compare our proposed methods with the most related methods from literature [3, 19, 24] and a baseline method for time series analysis [12, 16]. We outperform these in accuracy and computation time in our experiments. In the following we give further details about these methods.

The work by [3, 19] both present approaches to project 2-dimensional trajectories in the arc-angle space. Vlachos et al. [19] propose three different algorithms to calculate the angles: 'Exact', 'Relative' and 'cMass'. Their results showed the best performance for cMass, which we also observed in our experiments in Section 4. Feuerhake [3] apply their methods to identify similar trajectories of ball sport athletes. They split trajectories into disjunct segments, contrary to our sliding approach. As Procrustes analysis origins from analysing

shapes, we also compared our methods with the shape signature, a method often applied in shape recognition, as e.g. [24]. A shape signature is a representation of a multivariate time series as the one dimensional time series of the Euclidean distances of the original observations to a center point, typically the mean or gravity center.

In literature [12, 16] it is popular to apply the z-normalization before performing time series clustering or classification based on distance measures as e.g. the Euclidean distance or DTW. Section 4 demonstrates how this baseline method performs well on traditional time series, but struggles with recognizing similarities independent from rotation.

More recent works such as [1] focus on queries in databases, or [11] discuss different distance measures for trajectories, but they do neither discuss sliding approaches, nor how to cluster overlapping segments of trajectories. Also [18] compare different similarity measures for trajectories after scaling, shifting and rotating so that the first and final points of trajectories are fixed to each other. This step is not described detailed enough so that we could not include their method in our experiments.

Gawde and Pawar [4] present a method to represent a trajectory by multiple non-overlapping and non-intersecting polygons, which is hardly compatible with our sliding approach for the use case of analysing long trajectories as e.g. Fig. 10 shows. Apart from the polygon representation [4] also apply turning functions to represent the polygons in the arc-angle space, similar to [19], which we included in our experiments in Section 4. Yu et al. [22] elaborate the clustering of contemporary trajectories and therefore only define a similarity measure for trajectories whose recordings overlap in time, which clearly deviates from our methods. Vochten et al. [20] propose a generalized demonstration of motion trajectories in the field of robotics. They search for similar trajectories to a query trajectory by solving an optimal control problem, formulated as a non-convex NLP problem.

We propose a sliding algorithm that runtime-efficiently calculates the Euclidean distance after transformations to be rotation-scaling-shift invariant. Replacing the Euclidean distance by another distance measure would require to replace the Euclidean distance in the formulas (5), (8) and (9). Applying specifically DTW – DTW is a very popular distance metric for time series analysis – would limit the applicability of our proposed sliding algorithm, and so of the TSS clustering algorithm, since the runtime complexity of the Euclidean distance is $O(n)$, and for DTW it is between $O(n^2)$ and $O(n \times w)$, where n is the time series length and w is the window size parameter (see e.g. [15]). Further, when applying DTW, our proposed sliding algorithm would require adjusted logic for early abandoning similar to [8] or [13]. Finally the sliding algorithm would need major adjustment for integrating the rotation-scaling-shifting transformation and a runtime efficient incremental computation of the DTW, as e.g. presented by [10]. We conclude that in general our proposed method can be adjusted to integrate other time series distance methods instead, but see it out of the scope of this paper and leave it open for future work.

We summarize that our approach of sliding Procrustes analysis is novel and compared our methods to those most similar from literature.

7 Conclusion

We proposed a novel shift-scale-rotation invariant distance metric for motion trajectories based on Procrustes analysis. Further, we presented the novel algorithm S3KR, to slide the distance calculation along a longer trajectory and detect characteristic recurring motion patterns. Based on the distance metric and the sliding algorithm we also presented a novel time

series subsequence clustering algorithm based on spectral clustering. We demonstrated our methods to outperform the benchmark on synthetic data and a UCI benchmark trajectory data set, in performance (8% to 58%) and runtime (by a factor of up to 470). Also, we demonstrated the benefit of our proposed shift-scale-rotation invariant distance measure being a metric, by applying it in an M-Tree [2] and accelerating the search by a factor of more than 10 compared to the brute force search. Finally we applied our methods on GPS trajectories recorded when travelling by car, train, tram, or ship, to cluster these data and discover prototypical patterns for different transport modes. Our methods can be applied on spatial-temporal trajectory data from various domains for an enhanced understanding of motion content and to gain deep information about typical recurring maneuvers, and so support applications as activity recognition, transport mode identification, automated ticketing, analysis of athletes and many more.

Funding Open access funding provided by AIT Austrian Institute of Technology GmbH.

Data Availability The real world datasets analysed during the current study, and the R code to simulate the synthetic datasets are available here: Section 4 gives details to the original source of the AIS and UCI data. The Appendix gives further details for simulating the synthetic datasets.

Declarations

Conflicts of Interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix A: Trajectory simulation

For the experiments in Section 4 we simulate 2-dimensional trajectories. The goal was to simulate random walks that resemble motion patterns recorded by a GPS device carried by a person or vehicle in a real world scenario. The following formulas attempt that by defining the angle for updating the orientation of the trajectory dependent on its own past. The simulated trajectory x is then randomly modified by adding white noise, a random rotation, scaling and translation. The standard deviation σ_u for simulating the white noise u proved in experiments

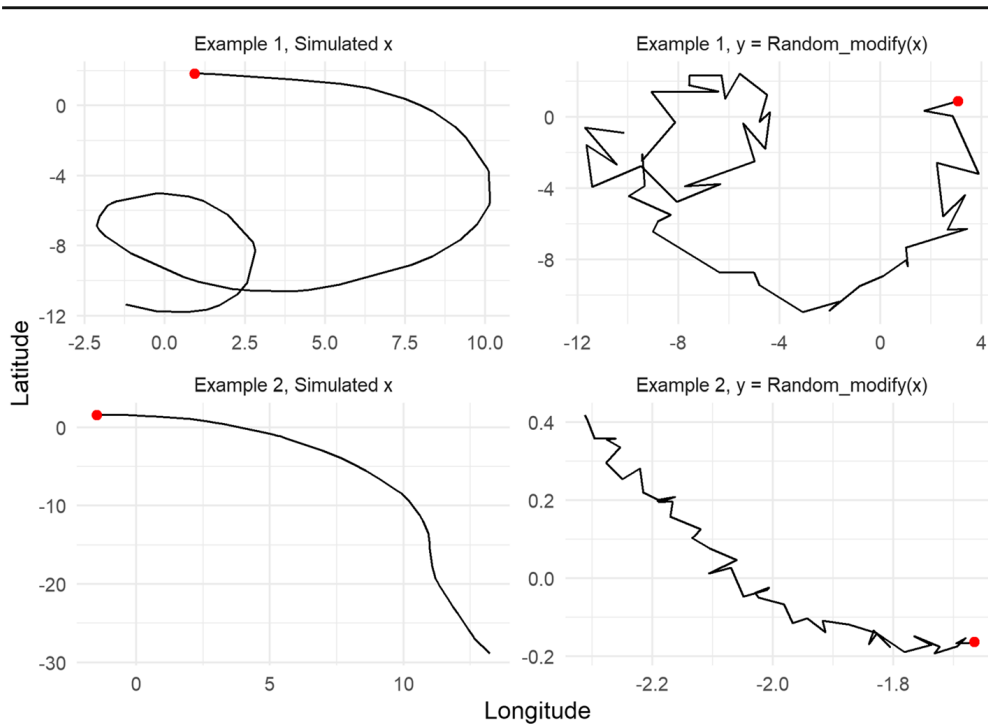


Fig. 12 Two examples of simulated trajectories. The upper left figure shows the first example and the random modified version on the upper right hand side. The second example is in the lower two figures. The red points indicate the initial positions of the trajectories

to be a crucial parameter, as Section 4.1 discusses.

$$\begin{aligned}
 x_0 &= \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad x_i = x_{i-1} + \begin{pmatrix} \cos \beta_i \\ \sin \beta_i \end{pmatrix} \cdot |\text{arclength}_i| \\
 \alpha_0 &= \beta_0 = 0 \\
 \alpha_i &= \alpha_{i-1} + z_i \\
 \beta_i &= \beta_{i-1} + \alpha_i \\
 y_i &= \begin{pmatrix} \cos \gamma & -\sin \gamma \\ \sin \gamma & \cos \gamma \end{pmatrix} \cdot \begin{pmatrix} x_{i1} + u_{i1} \\ x_{i2} + u_{i2} \end{pmatrix} \cdot |\text{scalefac}| + \begin{pmatrix} \text{trans}_1 \\ \text{trans}_2 \end{pmatrix} \\
 z_i, u_i, \text{arclength}_i, \text{trans}_i \text{ and scalefac} &\sim N(0, 1) \\
 u_i &\sim N(0, \sigma_u) \\
 \gamma &\sim U(1, 360)
 \end{aligned} \tag{17}$$

Figure 12 shows two examples of simulated trajectories, and their random modifications according to (17) with $\sigma_u = 0.6$.

Appendix B: δ^f is a metric

Theorem 1 δ^f is a semimetric, such that it fulfills for any x, y, z : $\delta^f(x, y) \geq 0$, $\delta^f(x, y) = \delta^f(y, x)$, and $\delta^f(x, y) \leq \delta^f(x, z) + \delta^f(z, y)$.

Proof Since the Euclidean distance d_E (induced by the Euclidean norm $\|\cdot\|$) is metric, also $\delta^f(x, y) \geq 0$ for any x and y . The symmetry follows directly from the definition. Let $x^* = \gamma(\tau(x))$:

$$\delta^f(x, y) \leq \delta^f(x, z) + \delta^f(z, y)$$

$$d_E(\underbrace{x^* \cdot R}_a, \underbrace{y^* \cdot S}_b) \leq d_E(\underbrace{x^* \cdot R}_a, \underbrace{z^* \cdot P}_c) + d_E(\underbrace{z^* \cdot P}_c, \underbrace{y^* \cdot S}_b).$$

Since a to c are in $\mathbf{R}^{n \times m}$, and independent from each other, and since d_E is metric, the triangle inequality holds for δ^f . \square

The last statement is not true for the distance function δ , since the rotation matrices S on the left side to rotate y to fit x , would be different from the rotation matrix on the right hand side to rotate y to fit z . So, compared to δ , δ^f has the major advantage of fulfilling the triangle inequality. Consequently δ^f is applicable to build database structures as e.g. an M-Tree [2], that enables fast k-NN and range queries⁹.

It follows from its definition that δ^f is an equivalence relation, and that in the space of all equivalence classes induced by δ^f , δ^f is metric. So $\delta^f(x, y) = 0 \Leftrightarrow x$ and y are equal after scaling, shifting and rotating to fit f .

References

1. Ali ME, Eusuf SS, Abdullah K, Choudhury FM, Culpepper JS, Sellis T (2018) The maximum trajectory coverage query in spatial databases. Proceedings of the VLDB Endowment 12(3):197–209
2. Ciaccia P, Patella M, Zezula P (1997) M-tree: an efficient access method for similarity search in metric spaces. In: Proceedings of the 23rd VLDB
3. Feuerhake U (2016) Recognition of repetitive movement pattern's the case of football analysis. ISPRS International Journal of Geo-Information 5(11):208
4. Gawde G, Pawar J (2018) Similarity search of time series trajectories based on shape. In: Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, pp 340–343
5. Gower JC, Dijksterhuis GB et al (2004) Procrustes problems, vol 30. Oxford University Press on Demand
6. Kabsch W (1976) A solution for the best rotation to relate two sets of vectors. Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography 32(5):922–923
7. Keogh E, Lin J (2005) Clustering of time-series subsequences is meaningless: implications for previous and future research. KAIS 8(2):154–177
8. Keogh E, Ratanamahatana CA (2005) Exact indexing of dynamic time warping. Knowledge and information systems 7(3):358–386
9. Leodolter M (2017) IncDTW: Incremental Calculation of Dynamic Time Warping. R package version 1(1):4
10. Leodolter M, Plant C, Brändle N (2021) IncDTW: an R package for incremental calculation of dynamic time warping. Journal of Statistical Software 99:1–23
11. Pelekis N, Kopanakis I, Marketos G, Ntoutsi I, Andrienko G, Theodoridis Y (2007) Similarity search in trajectory databases. In: 14th International Symposium on Temporal Representation and Reasoning (TIME'07), IEEE, pp 129–140
12. Rakthanmanon T, Campana B, Mueen A, Batista G, Westover B, Zhu Q, Zakaria J, Keogh E (2012) Searching and mining trillions of time series subsequences under dynamic time warping. In: 18th ACM SIGKDD, pp 262–270
13. Rath TM, Manmatha R (2002) Lower-bounding of dynamic time warping distances for multivariate time series. University of Massachusetts Amherst Technical Report MM 40:1–4
14. Reynolds AP, Richards G, de la Iglesia B, Rayward-Smith VJ (2006) Clustering rules: a comparison of partitioning and hierarchical clustering algorithms. Journal of Mathematical Modelling and Algorithms 5(4):475–504

⁹ also called ϵ -queries

15. Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans Acoust Speech Signal Process* 26(1):43–49
16. Schäfer P (2016) Scalable time series classification. *Data Mining and Knowledge Discovery* 30(5):1273–1298
17. Schubert E, Hess S, Morik K (2018) The relationship of DBSCAN to matrix factorization and spectral clustering. In: *Proceedings of LWDA*, pp 330–334
18. Toohey K, Duckham M (2015) Trajectory similarity measures. *Sigspatial Special* 7(1):43–50
19. Vlachos M, Gunopulos D, Das G (2004) Rotation invariant distance measures for trajectories. In: *Proceedings of the 10th ACM SIGKDD*, ACM, pp 707–712
20. Vochten M, De Laet T, De Schutter J (2019) Generalizing demonstrated motion trajectories using coordinate-free shape descriptors. *Robotics and Autonomous Systems* 122:103291
21. Williams BH, Toussaint M, Storkey AJ (2006) Extracting motion primitives from natural handwriting data. In: *ICANN*, Springer, pp 634–643
22. Yu Q, Luo Y, Chen C, Chen S (2019) Trajectory similarity clustering based on multi-feature distance measurement. *Applied Intelligence* 49(6):2315–2338
23. Zelnik-Manor L, Perona P (2005) Self-tuning spectral clustering. In: *Advances in neural information processing systems*, pp 1601–1608
24. Zhang D, Lu G (2004) Review of shape representation and description techniques. *Pattern recognition* 37(1):1–19
25. Zheng Y (2015) Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology* 6:1–41. <https://doi.org/10.1145/2743025>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Maximilian Leodolter received the MS degree in Mathematics in Economics from the TU Wien, in 2013. He worked as research fellow with the AIT Austrian Institute of Technology GmbH in the field of mobility data collection and analysis. Since 2017 he is a PhD candidate, developing algorithms for pattern recognition and clustering, supervised by Prof. Claudia Plant from the University of Vienna. Since 2020 he has been working as senior data scientist at Walter Group, where he is responsible for the development of AI solutions based on business data and spatiotemporal data.



Claudia Plant is full professor, leader of the Data Mining and Machine Learning research group at the Faculty of Computer Science University of Vienna, Austria. Her group focuses on new methods for exploratory data mining, e.g., clustering, anomaly detection, graph mining and matrix factorization. Many approaches relate unsupervised learning to data compression, i.e. the better the found patterns compress the data the more information we have learned. Other methods rely on finding statistically independent patterns or multiple non-redundant solutions, on ensemble learning or on nature-inspired concepts such as synchronization. Indexing techniques and methods for parallel hardware support exploring massive data. Claudia Plant has co-authored over 150 peer-reviewed publications, among them more than 30 contributions to the top-level data mining conferences KDD and ICDM and 4 Best Paper Awards. Papers on scalability aspects appeared at SIGMOD, ICDE, and the results of interdisciplinary projects in leading application-related journals such as Bioinformatics, Cerebral Cortex and Water Research.



Norbert Brändle received the PhD degree in computer science from the TU Wien, in 2002. He was lecturer and assistant professor with the Pattern Recognition and Image Processing Group at TU Wien, 1998-2003, and lecturer for multimedia information systems with the University of Applied Sciences Technikum Wien. He worked as a senior scientist with the AIT Austrian Institute of Technology GmbH coordinating projects in the field of mobility data collection and analysis. Since 2022 he works as a Senior Computer Vision Solution Architect at Crayon's Data & AI Center of Excellence Europe, where he is responsible for assessing customer needs and conceptualizing AI solutions. He is a member of the IEEE.